BIG DATA

AULA 6

Prof. Luis Henrique Alves Lourenço

CONVERSA INICIAL

Nesta aula, será apresentado a você um aprofundamento em alguns temas complementares aos estudos sobre *big data*. Inicialmente, serão apresentadas algumas tecnologias que podem não estar diretamente relacionadas ao Hadoop, mas que podem servir como alternativas muito interessantes e que podem ser encontradas no desenvolvimento de diversas aplicações de *big data* existentes. Em seguida, vamos explorar a abordagem de gerenciamento de dados armazenados em formato natural, conhecida como *data lake*. No tema seguinte, aprofundaremos o desenvolvimento de sistemas de recomendação utilizando *big data*. No quarto tema de nossa aula, conheceremos alguns detalhes sobre as plataformas de computação em nuvem. E, por fim, exploraremos os principais pontos no desenvolvimento de projetos de arquitetura de *big data*.

TEMA 1 – OUTRAS TECNOLOGIAS

Já conhecemos em detalhes muitas das principais tecnologias do ecossistema Hadoop, no entanto, podemos destacar mais algumas aplicações que podem ser muito úteis para o desenvolvimento de soluções de *big data*.

1.1 Impala

Impala é um motor de consultas SQL mantido pela Fundação Apache para executar em *clusters* Hadoop e que oferece a capacidade de atuar como banco de dados analítico massivamente paralelo. Entre as suas principais características destaca-se a possibilidade de realizar consultas SQL de baixa latência de dados armazenados em HDFS e HBase sem exigir movimentação ou transformação de dados. Essa ferramenta foi projetada para consultas analíticas em Haddop utilizando SQL ou ferramentas de *business inteligence* (BI). Com isso, o Impala é capaz de realizar processamento de dados em larga escala e consultas interativas no mesmo sistema através dos mesmos dados e metadados, ou seja, essas operações podem ser realizadas sem a necessidade de migrar conjuntos de dados para sistemas especializados ou formatos proprietários apenas para efetuar a análise dos dados.

A decisão de arquitetura do Impala de acessar os dados em HDFS ou HBase diretamente tem como consequência a redução de latência. Assim, é

possível obter desempenho muito maior do que aplicações semelhantes como o Hive, dependendo do tipo de consulta e configuração. Uma vez que o processamento é realizado localmente em cada nó, gargalos de rede são evitados. Isso permite que toda operação ocorra sem a conversão de formatos de dados e, com isso, todos os dados podem ser consultáveis sem atrasos.

1.2 Accumulo

O Accumulo é um projeto da Fundação Apache inspirado no sistema de armazenamento distribuído BigTable, do Google, ou seja, seu modelo de dados é baseado em armazenamento de chave-valor. O Accumulo foi projetado para utilizar o HDFS para armazenar os dados. Uma das principais características do Accumulo é a segurança em nível de célula, pois cada par chave-valor possui o seu próprio rótulo de segurança que tem a capacidade de limitar os resultados de uma consulta baseado nas autorizações de acesso do usuário. Dessa forma, é possível armazenar dados com diferentes requisitos de segurança na mesma tabela. Outra característica importante é o mecanismo de programação capaz de modificar pares chave-valor durante processos de gerenciamento de dados através do uso de iteradores (*SortedKeyValueIterators*), que são abstrações que permitem a implementação de operações diretamente no conjunto de servidores que armazena os dados (*TabletServers*). Desse modo, é possível realizar operações de transformações nos dados assim que são inseridos.

1.3 Redis

Redis é um sistema *open source* de armazenamento de dados em memória utilizado como banco de dados, cache e agente de mensagens (*message broker*). Seu nome é um acrônimo de *REmote DIctionary Server* (servidor de dicionário remoto). O Redis possui uma arquitetura mestre-subordinado que replica os dados de forma assíncrona por vários servidores. O sistema de armazenamento do Redis é baseado em uma estrutura de dados de chave-valor em que os dados se localizam na memória do servidor. Dessa forma, é possível reduzir o tempo de busca, uma vez que a quantidade de acessos a disco é reduzida. No entanto, o Redis também é capaz de fornecer o armazenamento de dados em disco como forma de garantir a persistência dos dados e sua rápida recuperação em caso de uma interrupção.

O Redis foi projetado inicialmente para ser um cache na memória para reduzir o tempo de acesso a bancos de dados, porém, suas características o fazem uma aplicação extremamente versátil. A estrutura de dados chave-valor permite que chaves façam o mapeamento para dados em diversos tipos, podendo ser dados de texto ou binários limitados a 512 MB. Além disso, os dados no Redis podem ser configurados com tempo de vida (TTL), ou seja, cada chave pode ser configurada para ser removida automaticamente depois de determinado período de tempo. Isso permite que o Redis seja bastante utilizado como gerenciador de sessões. Além disso, ele também é utilizado para a implementação de chat e sistemas de mensagens, filas, classificações em tempo real ou limitadores de taxas.

1.4 Ignite

Ignite é um projeto mantido pela Fundação Apache que pode ser definido como uma plataforma de computação em memória escalável, distribuída e tolerante a falhas para aplicações de tempo real que pode processar *terabytes* de dados em velocidade de memória. Uma de suas principais características é a capacidade de atuar tanto como uma camada de cache quanto como um sistema de armazenamento completo. Isso quer dizer que o Ignite é um sistema de computação em memória com a capacidade de armazenar dados em disco. Dessa forma, ele pode atuar como um banco de dados em memória, um *data grid* em memória ou como um banco de dados escalável horizontalmente. O Ignite pode ser entendido como um substituto do Redis, porém a principal diferença é que o Ignite possui características mais voltadas a um banco de dados, como a garantia das propriedades ACID, suporte à SQL.

1.5 NiFi

NiFi é um sistema de processamento e distribuição de dados que utiliza grafos dirigidos para automatizar e gerenciar fluxos de dados entre sistemas mantido pela fundação Apache. Dessa forma, o NiFi foi projetado para enfrentar os desafios que envolvem plataformas de processamento baseado em fluxos. Entre os seus pontos mais fortes, destacam-se o oferecimento de uma interface web para projetar, controlar e monitorar fluxos de dados. O NiFi permite configurar o nível de perda de dados, garantia de entrega, latência, vazão,

priorização dinâmica, alteração de fluxos, segurança, entre outras características. Alguns dos principais conceitos do Nifi incluem:

- ***FlowFile*:** é o pacote de informação (*Information Packet*) que representa os objetos de dados que se movimentam entre os sistemas.
- ***FlowFile processor*:** são os elementos processadores de fluxos de dados que realizam operações como roteamento de dados, transformação ou mediação entre sistemas.
- **Conexão:** são os elementos que fazem a ligação entre os processadores de fluxos de dados. Atua como uma espécie de *buffer* ou fila, que pode ser utilizada para priorizar certos dados dinamicamente.
- ***Flow controller*:** é um *scheduler* utilizado para definir a ligação entre os elementos do fluxo. Atua como um intermediário para facilitar a troca de *FlowFiles* entre os processadores de fluxos de dados.
- ***Process group*:** é um subconjunto de processadores de fluxos de dados e conexões capazes de receber dados e produzir resultados.

1.6 Ambari

Ambari é uma ferramenta que fornece uma interface Web fácil e intuitiva baseada em uma API REST para o provisionamento, gerenciamento e monitoramento de *clusters* Hadoop capaz de realizar a integração do Hadoop em uma infraestrutura já existente. Em sua interface, o Ambari exibe diversas ferramentas para monitorar o seu *cluster* Hadoop. Por exemplo, é possível monitorar a capacidade de armazenamento do *cluster*, a quantidade de nós ativos, a utilização de memória, rede e CPU, a carga do *cluster*, entre muitos outros detalhes. Além disso, o Ambari fornece uma interface para realizar a ativação, a desativação, o gerenciamento e utilização de diversos dos componentes de um cluster Hadoop. Para isso, o Ambari divide os componentes do Hadoop em três categorias:

- ***Core Hadoop*:** componentes básicos do Hadoop, como o HDFS e o MapReduce.
- ***Essential Hadoop*:** componentes projetados para facilitar o trabalho com os componentes do *Core Hadoop*. Inclui componentes como: Pig, Hive, HBase e Zookeeper.

- ***Hadoop Support*:** conjunto de componentes que permitem o gerenciamento e o monitoramento de uma instalação Hadoop e permite conectar o Hadoop a um *cluster*.

TEMA 2 – *DATA LAKE*

Já sabemos que a maior parte do tempo gasto em um projeto de análise de dados se concentra nas tarefas que envolvem o gerenciamento de dados tais como a identificação, limpeza e integração dos dados. Segundo Chessell et al. (2014), isso ocorre por três motivos principais:

- Os dados estão espalhados em diversas aplicações e sistemas de negócios;
- Os dados frequentemente precisam ser reorganizados e reformatados para serem mais facilmente analisados;
- Os dados devem ser atualizados frequentemente para manter a sua relevância.

Diferentemente de outras estratégias de armazenamento de dados como *data warehouse* e *data mart*, que consistem em dados extraídos de sistemas transacionais, ou seja, são formados por dados estruturados, como já sabemos, os dados que utilizamos no processamento de *big data* podem ser estruturados em diferentes formatos, semiestruturados ou nem precisam ser estruturados, pois são dados que podem vir de sistemas completamente diferentes decorrentes de necessidades muito diferentes. Portanto, esses modelos de armazenamento de dados não se mostram apropriados para tais aplicações. Por isso, é muito difícil a manutenção de sistemas de obtenção de dados para análise e, portanto, se mostra muito necessária para a criação de uma estrutura de governança de dados que possa incorporar dados processados ou não de forma que possam ser utilizados por ferramentas de *big data* para a geração de informações valiosas através da análise de tais dados.

O termo *data lake* foi inventado e descrito pela primeira vez por James Dixon em seu blog, em 2010, da seguinte forma:

> Se você pensar em um *data mart* como uma loja de garrafas de água (água limpa, embalada e estruturada para ser consumida facilmente), o *data lake* seria um corpo de água maior em estado natural. O conteúdo do *data l*flui a partir de uma fonte que preenche o lago e vários usuários do lago podem vir e examinar, mergulhar ou colher amostras. (Dixon, 2010)

Uma vez que se trata de dados em seu estado natural para uma comunidade de usuários, o conceito de *data lake* se mostra muito mais adequado a soluções de *big data*, nas quais um *data lake* corresponde a um conjunto de dados de diversas fontes que são armazenados sem que sejam transformados e onde os dados só serão transformados e terão esquemas aplicados para atender às necessidades de sua análise. Um *data lake* pode, então, receber quaisquer dados e seus dados podem ser livremente atualizados.

No entanto, sem um gerenciamento adequado, os dados podem se degradar a um estado onde não existe mais como saber a sua confiabilidade, fazendo com que percam seu valor e o *data lake* se torne em um tipo de pântano (*data swamp*) que não permite mais a extração de informações úteis. Para que o valor dos dados contidos em um *data lake* não seja perdido, faz-se necessário criar uma solução de *data lake* que inclua as noções de gerenciamento, acessibilidade e governança. Para muitos, essa solução é uma adaptação de *data lake* chamada de *data reservoir* (ou reservatório de dados) (Chessell et al., 2014). Um *data reservoir* pode ser compreendido como um *data lake* capaz de catalogar e proteger os seus dados.

Dessa forma, a estratégia de armazenamento de dados *data lake* pode oferecer uma redução de complexidade, que as estratégias *data warehouse* e *data mart* não são capazes de oferecer, uma vez que, para atender à mesma demanda que um único *data lake* é capaz de atender, faz-se necessária a utilização de diversos desses sistemas. Isso também gera impacto no custo e eficiência desses sistemas. Além disso, *data lakes* com bons esquemas de governança são capazes de trazer mais transparência para o uso dos dados.

Um dos principais conceitos que envolvem a governança de dados é o conceito de metadado que, em outras palavras, são as informações que se podem obter a respeito dos dados em si. Ou seja, são os dados que os sistemas de *data lake* utilizam para gerenciar os seus dados e permitem a governança de dados. Os metadados podem ser divididos em três categorias:

- **Metadados técnicos:** são os metadados que fornecem informações a respeito da forma ou estrutura dos dados, como tamanho, tipo de dado, esquema.
- **Metadados operacionais:** são os metadados que fornecem informações sobre a cadeia de fornecimento dos dados como qualidade, perfil,

proveniência dos dados, e a sua linhagem, ou seja, sua cadeia de fornecimento.

- **Metadados de governança (ou de negócio):** são os metadados que fornecem a terminologia de negócio de cada dado, ou seja, são as informações a respeito do que os dados representam para o usuário final e que podem ser utilizadas para tornar mais fácil encontrar e entender tais dados, como *tags*, descritores, nomes de negócios, qualidade e regras de mascaramento.

Uma vez que os metadados são armazenados, eles permitem a possibilidade de buscar, localizar e aprender sobre os dados disponíveis no *data lake*. Para isso, é necessário gerenciar uma camada adicional para armazenar e catalogar os metadados. Essa é a primeira garantia que impede que o seu *data lake* se torne um *data swamp*. Dessa forma, é necessário que todos os dados que sejam inseridos no *data lake* passem por um processo de criação e extração de metadados.

2.1 Maturidade de *data lake*

A importância de se trazer dados em seu formato natural para o sistema de armazenamento é que esses dados podem ser utilizados para análise de autoatendimento (*self-service analytics*), que é uma abordagem de análise mais avançada, em que os consumidores dos dados possuem a habilidade de explora-los diretamente. Essa análise possui grandes desafios no que se refere à governança e à segurança de dados. A governança da informação de um projeto de *big data* (ou seja, a atitude tomada em relação ao gerenciamento de seus dados) se manifesta diretamente nos resultados obtidos por ele, uma vez que permitem garantir segurança, organização, catálogo das informações e acessibilidade aos dados. No entanto, o grande volume que o *big data* pode atingir nos mostra que não é prático aplicar um conjunto rígido de processos sobre todo o conjunto de dados. Para isso, Gorelik (2019) define alguns estágios de maturidade de um *data lake*:

- ***Data puddle* (ou poça de dados)**: estágio inicial onde os dados de um *data mart* são utilizados em conjunto com tecnologia *big data*. Normalmente, é o primeiro passo na adoção de uma estratégia de Big Data dentro de uma organização. Essas estruturas possuem um escopo

limitado e possuem o propósito de atender às necessidades de um time ou um projeto específico, muitas vezes servindo para atender a uma demanda por processos que exigem um uso intenso de computação e dados. Ou ainda servindo de local de testes para cientistas de dados. Sua construção e manutenção exigem um alto envolvimento de uma equipe técnica.

- ***Data ponds* (ou lagoa de dados)**: estágio em que a organização possui uma coleção de data *puddles*, ou seja, na prática funciona como um *data warehouse* mal projetado, com diversos *data marts* para propósitos específicos, que utilizam tecnologia *big data*, porém, comparado com *data puddles*, utilizam tecnologias mais baratas e escaláveis que *data warehouses* e *data marts* relacionais. Não possuem muitas vantagens, uma vez que continuam com limitações semelhantes aos *data puddles* e não colaboram para melhorar o acesso aos dados, nem para a tomada de decisões de negócio baseadas em dados.
- ***Data lake* (ou lago de dados)**: difere do *data pond* no sentido em que suportam análise de autoatendimento, ou seja, os usuários são capazes de realizar buscas e localizar dados diretamente. Além disso, são capazes de armazenar dados que não possuem propósito em nenhum projeto atual, mas que podem ser úteis para usuários de negócios.
- ***Data ocean*:** o oceano de dados expande o conceito de *data lake* para toda uma organização. Dessa forma, todos os dados da organização estão disponíveis para análise de autoatendimento e decisões estratégicas de negócios baseados em dados.

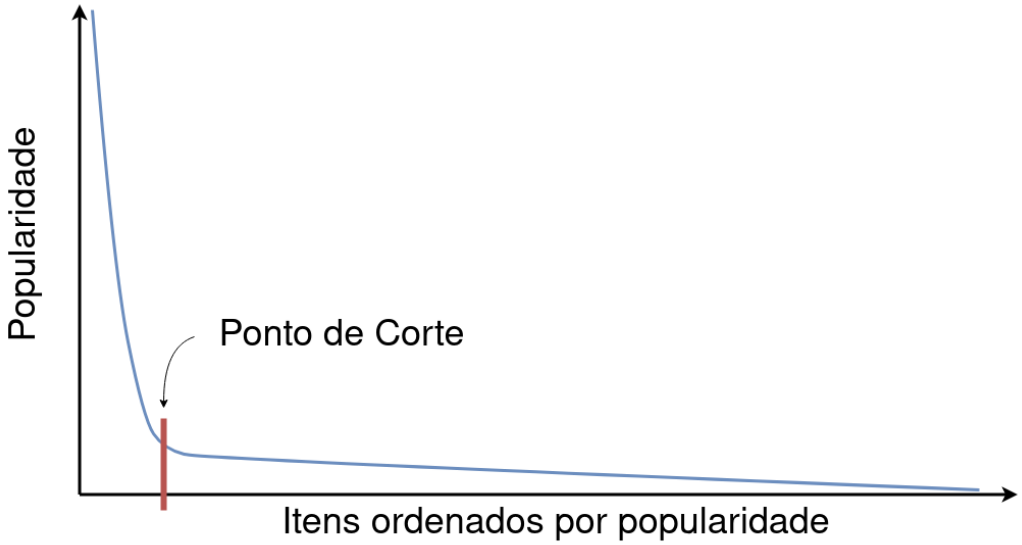
TEMA 3 – SISTEMAS DE RECOMENDAÇÃO

Sistemas de recomendação são uma classe de aplicações que buscam prever a reação dos usuários em relação a um conjunto de opções. Dessa forma, esses sistemas são capazes de fornecer uma experiência personalizada para cada usuário, muitas vezes com o intuito de oferecer opções mais adequadas. Para isso, esses sistemas buscam encontrar relações entre suas distintas partes, seja por meio da comparação de ações passadas de forma a permitir uma predição de quais são as ações futuras mais prováveis, seja por meio da comparação entre as características de cada elemento.

Atualmente, qualquer serviço que ofereça acesso a conteúdo pode possuir um catálogo imenso. Em muitos casos, os usuários já sabem de antemão o que desejam. Dessa forma, o usuário pode acessar o conteúdo por meio de uma simples busca pelo catálogo, porém, em outros casos, os usuários não sabem exatamente o que querem e é aqui que os sistemas de recomendação se tornam necessários. Assim, o sistema recomenda ao usuário certos produtos que ele acredita ser de interesse baseado nas informações que ele sabe do usuário. Então, podemos dizer que a necessidade de sistemas de recomendação surgiu devido à dificuldade em divulgar catálogos imensos de forma mais eficiente. O desafio aqui pode ser explicado através do fenômeno da cauda longa (*the long tail*). Esse fenômeno é baseado em uma característica conhecida de algumas distribuições estatísticas que diz que a maioria das ocorrências correspondem a uma minoria dos itens na distribuição.

Lojas físicas possuem uma limitação de quantos produtos podem possuir em suas prateleiras, por isso precisam escolher quais produtos deseja expor para seus usuários. Nesse cenário, a recomendação é bastante simples: uma vez que o vendedor não é capaz de conhecer suficientemente a todos os usuários, a escolha de quais produtos devem ser expostos se baseia em números agregados. Ou seja, para maximizar suas vendas, o lojista deve escolher expor os produtos mais populares em suas prateleiras. Dessa forma, o fenômeno da cauda longa é capaz de distinguir sistemas limitados como os das lojas físicas dos sistemas on-line que não possuem uma limitação de espaço para exibir seus produtos. Na Figura 1, o gráfico representa o fenômeno da cauda longa em que o eixo vertical representa a popularidade (a quantidade de vendas) dos produtos, representados de forma ordenada no eixo horizontal. Nele podemos observar que o volume de vendas está concentrado em uma parte reduzida dos produtos e que há um ponto de corte onde a partir do qual não há viabilidade para o lojista explorar. Essa parte da curva é denominada de *cauda longa*.

Figura 1 – Cauda longa



Fonte: Leskovec; Rajaraman; Ullman, 2010.

Por outro lado, os sistemas on-line não possuem tal limitação e podem explorar toda a faixa de itens. Por essa razão, o catálogo de sistemas on-line costuma ser muitas vezes maior que o catálogo de lojas físicas. No entanto, a existência de tantos itens é um desafio para qualquer usuário encontrar os itens de seu interesse. Dessa forma, podemos dizer que mais opções de escolha exigem melhores filtros. Além disso, vale a pena observar que esses sistemas não são úteis apenas para sistemas de venda ou recomendar músicas, filmes e podem ser utilizados em redes sociais para selecionar a lista de postagens que aparecerá no *feed* de cada usuário e, até mesmo, os motores de busca podem funcionar como sistemas de recomendação, uma vez que passem a retornar buscas direcionadas a determinado perfil de usuário. Esses sistemas têm em comum a característica de proporcionar ao usuário uma experiência personalizada, ou seja, eles levam em consideração a relação entre as características ou comportamentos do usuário em relação aos itens com que interage.

3.1 Tipos de recomendadores

Os sistemas de recomendação podem ser divididos em grupos baseados em como são construídos:

- **Editoriais ou curadorias:** são as listas de recomendações criadas à mão. Normalmente, levam em consideração os interesses de quem está criando, por exemplo, uma lista de favoritos. Não levam em consideração nenhuma característica do usuário.
- **Agregações simples:** listas que agregam alguma característica do próprio conteúdo. Por exemplo: *Top 10*, listas de popularidade, mais recentes. Essas listas podem até levar em consideração algum fator relacionado com seus usuários, por exemplo, a soma de interações que a totalidade de usuários tem com o conteúdo (vendas, *views* etc.).
- **Recomendadores individualizados:** são as recomendações que levam em conta as características dos usuários e dos produtos para gerar uma experiência personalizada para cada usuário. Aqui podemos elencar os sistemas de recomendação que nos interessam nesse tema.

Os recomendadores individualizados utilizam uma função de utilidade que recebe um usuário e um item e os mapeia a uma avaliação. Dessa forma, é possível criar uma matriz de utilidade com a avaliação que cada usuário faz de cada item. Obviamente trata-se de uma matriz esparsa, uma vez que, na maioria dos casos, os usuários avaliam apenas uma parcela muito pequena do conjunto de itens disponíveis. Existem duas formas de se obter avaliações de usuários. A primeira delas são as avaliações explícitas, que é quando o usuário ativamente determina uma nota de avaliação para o item. No entanto, apenas uma minoria dos usuários realiza avaliações, logo não escalam muito bem. A outra forma são as avaliações implícitas em que o sistema determina as avaliações de acordo com o comportamento do usuário.

Uma vez que temos avaliações que representam a relação entre cada usuário e cada item, podemos obter a matriz de utilidade. Ela é muito importante para os sistemas de recomendação, pois as recomendações se baseiam nas previsões de preenchimento das avaliações desconhecidas. Existem duas principais abordagens para isso: recomendações baseadas em conteúdo (*content-based recommendations*) e filtragem colaborativa (*collaborative filtering*).

3.2 *Content-based recommendations*

A ideia principal dessa abordagem é recomendar ao usuário itens parecidos, ou seja, com características semelhantes aos que ele avaliou. Dessa forma, o foco dessa abordagem está nas características dos itens. A similaridade entre diferentes itens está baseada na similaridade de suas características. Portanto, é importante construir um perfil para cada item que represente suas características e possa ser comparado para fins de calcular a similaridade com outros perfis de itens. É conveniente pensar em um perfil como um vetor de características. Da mesma forma, o perfil de um usuário é construído baseado em suas interações com os itens. Podemos pensar que os perfis de usuários são calculados por uma soma ponderada (levando em consideração a avaliação do usuário) dos perfis de itens avaliados. Assim, é possível realizar predições de avaliação, ou seja, é possível inferir qual será a avaliação de um item para determinado usuário através de seus perfis.

As vantagens de usar essa abordagem incluem a não dependência entre usuários, o que significa que não são necessários dados de outros usuários para recomendar itens a um usuário. Isso também indica que é possível fazer recomendações para usuários com gostos específicos. Além disso, não são necessárias avaliações prévias. Um usuário pode receber recomendações de um item novo assim que ele é inserido e também permite que a recomendação possa ser explicada, uma vez que se conhecem as características que levaram até ela.

As desvantagens incluem a dificuldade em encontrar as características adequadas. Não é uma tarefa simples definir quais são as melhores características para descrever um tipo de item. Outra desvantagem é a superespecialização, ou seja, itens fora do perfil de conteúdo do usuário nunca são recomendados. Múltiplos interesses são ignorados, assim como não levam em consideração a popularidade desses itens. Por fim, existe a dificuldade de criar perfis para novos usuários, uma vez que estes não avaliaram nenhum item.

3.3 *Collaborative filtering*

A abordagem de filtragem colaborativa (ou *collaborative filtering*) parte do pressuposto de que usuários com avaliações semelhantes podem ser utilizados para estimar a avaliação que um deles fará de determinado item. Para

isso é necessário calcular a similaridade entre os usuários, ou seja, o quanto as suas avaliações são parecidas. Algumas estratégias podem ser utilizadas para isso, porém a que apresenta os melhores resultados é a *Pearson Correlation*. Essa forma de calcular a similaridade entre dois perfis de usuários utiliza o cálculo do cosseno entre dois perfis normalizados pela subtração da média para cada usuário. Dessa forma, a avaliação média de cada usuário se torna zero, eliminando a distorções. Levando-se em conta o cálculo de similaridades de perfis de usuários, podemos realizar a predição de avaliações para dado usuário através da média das avaliações do conjunto de usuários mais semelhantes a ele ponderada pela soma das similaridades do conjunto de usuários semelhantes.

Podemos aplicar o mesmo princípio para as avaliações entre itens (ou *Item-Item collaborative filtering*). Dessa forma, para estimar a avaliação que determinado usuário poderá fazer para determinado item, deve-se buscar por itens avaliados pelo usuário com avaliações semelhantes. Para isso, as mesmas funções de métrica e predição do modelo de avaliação entre usuários (*User-User collaborative filtering*). Ambos os modelos de avaliação possuem abordagens complementares. Na prática, o modelo de avaliação entre itens possui melhores resultados para a maioria dos casos, pois itens são relativamente mais simples que usuários, portanto, a similaridade de itens é mais significativa que a de usuários.

Além disso, a grande vantagem da abordagem de filtragem colaborativa é que pode ser aplicada a qualquer tipo de item e não é necessária nenhuma seleção de características. Enquanto que o ponto fraco de tal abordagem é a necessidade de uma quantidade mínima de usuários para comparar similaridades. Os dados são esparsos, ou seja, é difícil encontrar usuários que avaliaram os mesmos itens e não é possível recomendar um item que nunca foi avaliado. Por fim, temos de considerar o viés de popularidade, ou seja, a tendência de que os itens mais populares sejam mais recomendados por serem normalmente melhor avaliados.

3.4 Métodos híbridos

É possível adicionar métodos de uma abordagem em outra abordagem, por exemplo, ao adicionar métodos baseados em conteúdo (*content-based*) em uma filtragem colaborativa. Dessa forma, é possível criar perfis de item para

resolver o problema de novos itens que nunca foram avaliados em uma filtragem colaborativa. Além disso, é possível implementar mais de um recomendador diferente e combinar as suas predições utilizando, por exemplo, abordagens que empregam modelos lineares, por exemplo, o uso de informações demográficas para lidar com o problema de perfis de novos usuários.

3.5 Avaliação de sistemas de recomendação

Sistemas de recomendação podem ser avaliados por meio da remoção de uma parte do conjunto de dados da matriz de utilidade. Dessa forma, os valores removidos são calculados pelo sistema de recomendação. Em seguida, os dados estimados podem ser comparados com os dados conhecidos que foram removidos da matriz de utilidade por meio do cálculo da raiz do erro médio quadrático (RMSE, em inglês). A RMSE é uma medida muito utilizada para aferir a qualidade de um modelo e funciona como uma medida análoga ao desvio padrão, sendo capaz de explicitar a diferença entre a predição e o valor real.

TEMA 4 – COMPUTAÇÃO EM NUVEM

Como vimos em nossos estudos, a metodologia de processamento de *big data* permitiu a captura, o armazenamento e a análise de imensas quantidades de dados. Além disso, a arquitetura Hadoop permitiu a implementação de clusters em máquinas comuns. Com isso, podemos analisar dados de muitas fontes que antes eram ignoradas como *logs*, dados de sensores, informações de redes sociais e muitos outros dados que não se encontram armazenados em bancos de dados relacionais. A análise de dados não está mais limitada aos dados estruturados, e agora pode incluir um volume de dados cada vez maior. Aliado a isso, a tecnologia dos equipamentos de armazenamento de dados continua evoluindo e, com isso, os preços do *gigabyte* armazenado seguem a tendência de queda. No entanto, o volume de dados sendo produzidos e que podem ser incluídos para análise tem crescido tão rápido que mesmo todo esse desenvolvimento de tecnologias mais baratas para processar e armazenar cada vez mais dados não tem sido suficiente para que muitas empresas pequenas e médias possam adotar essas tecnologias.

Dessa forma, surge como uma opção viável para essas empresas a adoção de sistemas de *cloud computing* (ou computação em nuvem) para

armazenar e processar seus dados utilizando tecnologia de *big data*. Nesse contexto, *cloud computing* representa o acesso de rede sob demanda a recursos computacionais fornecidos por uma entidade externa. Ou seja, a *cloud computing* permite o acesso a serviços de computação escaláveis e elásticos por meio da internet. Esses serviços podem oferecer às empresas uma redução de infraestrutura, bem como uma redução nos custos com licenças e manutenção de softwares específicos, além de permitir um ambiente viável para a implementação de tecnologias que fazem uso do processamento de *big data* em empresas de pequeno e médio porte. Também devemos destacar que esses sistemas podem servir como forma de testar novas tecnologias antes de a empresa optar por fazer um investimento maior. Por outro lado, a preocupação com a segurança e o controle dos dados são pontos de discussão.

Quando uma empresa investe em uma infraestrutura de *cloud computing* interna à própria organização, essa infraestrutura é classificada como privada, em contraposição às infraestruturas de *cloud computing* públicas, que são aquelas abertas ao uso por organizações externas.

4.1 Modelos de serviço de *cloud computing*

Uma vez que a tecnologia de *cloud computing* se propõe a oferecer serviços para atender às demandas de computação de entidades externas, podemos classificar os serviços oferecidos em categorias como:

- **PaaS (*platform as a service*):** a plataforma como um serviço pode ser entendida como o uso de *cloud computing* para oferecer plataformas para o desenvolvimento e uso de aplicações customizadas necessárias para analisar grandes quantidades de dados. Ou seja, são aplicações que oferecem soluções para o design de aplicações, testes de aplicação, versionamento, integração, hospedagem, gerenciamento de estado, e demais ferramentas de desenvolvimento. Esse tipo de serviço pode reduzir os requisitos para o desenvolvimento de novos sistemas, além de permitir uma redução de custos e riscos.
- **SaaS (*software as a service*):** software como um serviço fornece às empresas aplicações que são armazenadas e executam em servidores virtuais. O custo de um serviço normalmente é cobrado por largura de banda, tempo de uso e número de usuários, e as empresas se beneficiam

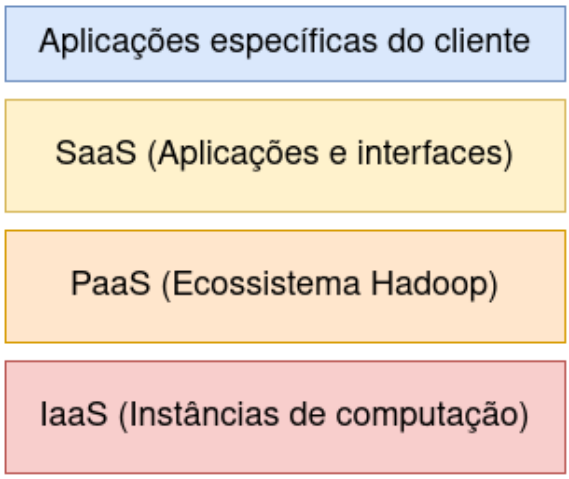
de menores esforços de administração e manutenção das aplicações, além da facilidade de acesso ao software em qualquer lugar. A principal diferença para o PaaS é que o modelo SaaS não oferece uma solução customizada enquanto que no modelo PaaS a empresa pode desenvolver soluções específicas às suas necessidades.

- **IaaS (*Infrastructure as a Service*):** a infraestrutura como um serviço fornece acesso ao uso de equipamentos para recursos computacionais como armazenamento, servidores, dispositivos de rede e outros tipos de hardware. Portanto, podemos afirmar que, de certa forma, todos os fornecedores de serviços de *cloud computing* o fazem sobre uma nuvem de IaaS. Além disso, esse modelo é muito utilizado para oferecer os serviços de recuperação de desastres, serviços de computação, serviços de armazenamento, serviços de *data center*, infraestrutura de *desktop* virtual, entre outros.

4.2 *Big data as a service*

Os modelos de *cloud computing* descritos anteriormente podem ser visualizados como camadas em um modelo de negócios. Em tal modelo, a camada mais básica seriam todos os serviços providos por um modelo IaaS, que incluem as instâncias de computação, hardware ou máquinas virtuais, armazenamento, rede e toda infraestrutura necessária. Em uma camada intermediária seriam encontrados os serviços de um modelo PaaS, ou seja, em tal camada se encontram as aplicações que fornecem uma plataforma para a execução de um software customizado para as necessidades do cliente tais como serviços de banco de dados, gerenciamento de cluster, distribuição e gerenciamento de processamento, entre outras aplicações. E em uma camada superior se encontram os serviços de um modelo SaaS, que são as aplicações genéricas que são utilizadas pelo usuário através de APIs, interfaces web.

Figura 2 – *Big data as a service*



Como podemos observar na Figura 2, um serviço de *big data* implementado sobre uma pilha de camadas de *cloud computing* naturalmente se encontra na camada de PaaS, uma vez que consideramos o Hadoop, ou qualquer outra tecnologia de processamento e armazenamento distribuído, como uma plataforma onde soluções são implementadas. Dessa forma, temos variações de BDaaS (*Big Data as a Service*) que podem implementar a camada de IaaS, a camada de SaaS ou ambas na mesma nuvem.

4.3 Principais fornecedores de BDaaS

Podemos destacar algumas das principais empresas no mundo que fornecem serviços de *big data* por meio de serviços públicos de *cloud computing*:

- **Amazon EMR:** o Amazon Elastic MapReduce é a ferramenta de *big data* da plataforma Amazon Web Services (AWS) que implementa as camadas IaaS e PaaS. Atualmente é a maior plataforma de *big data* em nuvem. O EMR é baseado no Hadoop e permite utilizar ferramentas como Spark, Hive, HBase, Flink, Presto, entre muitos outros serviços. O EMR processa big data utilizando clusters Hadoop hospedados nos servidores virtuais da nuvem pública Amazon Elastic Compute Cloud (EC2). Além disso, ele implementa uma infraestrutura de armazenamento distribuída própria, a Amazon Simple Storage Service (S3), que substitui o HDFS.
- **Google Cloud Dataproc:** é a ferramenta de *big data* do Google, que fornece serviços Spark e Hadoop, além de componentes como Hive, HBase, Zeppelin, Zookeeper, Presto, Pig, entre outros. Implementa uma

combinação entre as camadas IaaS e PaaS. O Cloud Dataproc está integrado com outros serviços do Google Cloud Platform, como BigQuery, Cloud Storage, Cloud Bigtable, Stackdriver Logging e Stackdriver Monitoring.

- **Microsoft Azure:** implementa uma integração entre IaaS e PaaS, que fornece serviços hospedados nos servidores virtuais de *cloud computing Azure virtual machines*. O Azure também implementa serviços de armazenamento (*Azure blob storage*), CDN, serviço de *containers* baseados em *docker* (*Azure container services*), processamento em lote (*Azure batch*), computação sem servidor (*Azure functions*), e um serviço para permitir o uso e gerenciamento de *clusters* Hadoop e Spark (*Azure HDInsight*).

TEMA 5 – DESIGN DE ARQUITETURA *BIG DATA*

Uma vez que conhecemos os conceitos que envolvem a arquitetura *big data* assim como as aplicações que implementam suas características, podemos pensar em como combinar todas essas tecnologias e ideias para o desenvolvimento de um produto visando atender a alguma necessidade existente. Devemos considerar também o impacto que tais tecnologias estão gerando na sociedade e nas empresas. Atualmente aplicações baseadas em *big data* representam um papel chave no ponto de vista dos negócios. No entanto, o desenvolvimento de aplicações de *big data* enfrenta dificuldades maiores que o desenvolvimento de sistemas baseados em tecnologias tradicionais. Esses desafios podem surgir da dificuldade de selecionar quais tecnologias de *big data* devem ser empregadas para cada tipo de necessidade. Além disso, também existem desafios no que se trata da complexidade em integrar sistemas de *big data* com os sistemas tradicionais existentes. Isso se expressa em algumas estimativas que dizem que mais da metade dos projetos de *big data* não conseguem ser postos em prática. As soluções de *big data* muitas vezes buscam objetivos como otimizar processos de negócios, adquirir vantagem competitiva, otimizar operações, entre muitos outros. Portanto, ao não conseguir concluir esses projetos, as empresas falham em obter uma grande variedade de resultados positivos.

No entanto, devido à complexidade de tal tarefa, não existe uma única maneira de implementar uma aplicação de *big data*. Dessa forma, precisamos

das informações que serão utilizadas para guiar o desenvolvimento. Uma das maneiras de se obter tais informações é o entendimento detalhado das necessidades que a aplicação busca atender, ou seja, precisamos compreender questões tais como **quem serão os usuários**, **quais são os problemas** que a aplicação deve resolver, **quais os benefícios** mais importantes para os usuários, **como garantir** que sabemos o que os usuários querem e precisam e **como é a experiência** para os usuários. Dessa forma, é possível obter os requisitos-chave para o desenvolvimento.

Uma das abordagens que pode auxiliar na tarefa de criar uma boa documentação que seja capaz de levantar os requisitos corretos para o desenvolvimento de uma aplicação de *big data* evitando desperdícios e garantindo que a tudo que está sendo implementado esteja alinhado com as necessidades do usuário é a abordagem de desenvolvimento de produtos da Amazon conhecida como *Working Backwards*, ou trabalhando de trás para frente. Esse método é utilizado na empresa para o desenvolvimento de novos produtos focando no ponto de vista do usuário. Dessa forma, é possível obter o melhor entendimento sobre as necessidades do usuário e gerar documentação durante o levantamento de requisitos e antes de iniciar o desenvolvimento. Assim, todas as decisões de projeto acabam sendo tomadas de forma a otimizar a solução do problema real.

A ideia dessa abordagem consiste em começar pelo anúncio do produto, ou seja, pela divulgação de um *Press Release*. Apesar do nome, esse anúncio não precisa ser um comunicado de imprensa e vai depender do contexto de sua aplicação. Em muitos casos pode ser apenas uma comunicação interna da empresa para o desenvolvimento de uma ferramenta que será utilizada apenas internamente. O importante é que o produto seja anunciado na primeira fase para que os documentos de requisitos reflitam todas as críticas que o anúncio receber. Só depois que os requisitos estiverem sido obtidos é que podemos começar a fase de projetar o produto que será, então, desenvolvido. Dessa forma, o produto final tende a ser melhor ajustado às necessidades dos usuários.

5.1 Requisitos de aplicações *big data*

Quando estamos analisando os requisitos dos usuários em relação a uma aplicação de *big data* temos que considerar diversos fatores. Primeiro, é importante destacar que devemos levar em consideração a infraestrutura e a

experiência da equipe. É uma boa estratégia utilizar o conhecimento que a equipe já possui em vez de buscar uma tecnologia que seja nova para toda a equipe. Muitas vezes vale mais a pena fazer uma substituição de tecnologia para poder aproveitar algo que a equipe já possui afinidade do que a solução ideal, porém desconhecida pela equipe.

Como sabemos, um dos principais aspectos a se considerar em *big data* é o tamanho dos dados que o produto deverá manipular. Nesse ponto, o *Working Backwards* nos lembra que todas as decisões devem ser tomadas tendo como foco o ponto de vista do usuário. Dessa forma, não devemos utilizar uma tecnologia de *big data*, como o armazenamento distribuído se o volume de dados previsto não é tão grande, ou seja, não devemos tomar decisões de desenvolvimento com base na tecnologia que queremos adotar, mas devemos optar pela tecnologia que melhor atende ao cenário previsto. Deve-se tomar especial atenção ao tentar prever os requisitos para cenários futuros. No caso do armazenamento de dados, pode ser muito caro mover os dados para outro lugar, uma vez que o local de armazenamento foi definido. Além disso, é preciso definir quais serão as formas de ingestão de dados, ou seja, quais as tecnologias disponíveis para que os dados possam ser incorporados ao sistema.

Vale a pena destacar que a solução mais simples normalmente é a que vai atender melhor aos requisitos da aplicação. Escalar a complexidade de um sistema para atender a uma demanda além do previsto não é uma boa estratégia. Você pode aumentar os custos de manutenção sem que haja demanda para tal. Além disso, várias das tecnologias em *big data* são intercambiáveis e, dessa forma, podem ser substituídas facilmente. Tente sempre avaliar qual é o mínimo de infraestrutura necessária para operar a aplicação.

É importante lembrar que, em alguns casos, os dados devem ser mantidos armazenados no sistema por diversos motivos, como uma necessidade de realizar auditorias, ou por necessidades jurídicas, como é o caso de muitas aplicações financeiras. Dessa forma, pode ser necessária a implementação de algum tipo de política de retenção de dados, como é o caso de dados de navegação ou dados sensíveis de usuários. É interessante que sejam eliminados eventualmente por motivos de privacidade. Em casos como esses, é interessante haver uma avaliação prévia antes da implementação de uma estratégia de armazenamento de dados, para que os requisitos avaliados

sejam atendidos pela política de governança dos dados. Além disso, também é importante definir uma política de expurgo, ou seja, procedimentos para a remoção de dados que não sejam mais necessários para nenhum propósito.

Também é importante ressaltar que as medidas de segurança adequadas devem ser tomadas para proteger os dados dos usuários. Vale sempre a pena verificar com o departamento jurídico da empresa se as regulamentações de segurança estão sendo cumpridas em relação à legislação de todos os países onde a aplicação estará em operação.

Outro aspecto que podemos levar em conta é a relação do sistema com o teorema CAP. Devemos avaliar quais as características do teorema que devem ser priorizadas, uma vez que o próprio teorema define que é impossível garantir os três atributos (consistência, disponibilidade e tolerância a partições) simultaneamente. A falta de disponibilidade em alguns sistemas pode custar muito dinheiro ou ser críticas a determinada aplicação. Para esses casos, deve se optar por um sistema de armazenamento que garanta que cada requisição por dados deve receber uma resposta, mesmo que não seja possível garantir que a resposta contém o dado atualizado. Da mesma forma, um sistema que depende que os dados requisitados sejam sempre os mais recentes, ou seja, que as transações não se percam e sejam executadas em uma ordem específica, devem priorizar soluções de armazenamento que garantam a consistência.

Outra questão muito importante trata dos padrões de acesso dos usuários em relação ao sistema. Neste ponto, precisamos entender qual é o tempo que o usuário precisa para ser atendido pela aplicação. É necessário avaliar quais soluções podem atender o usuário com um tempo de resposta compatível com sua expectativa. Por exemplo, para sistemas que precisam responder a requisições de usuários em milissegundos, pode ser necessária a adoção de bancos NoSQL como HBase ou Cassandra.

Os dados armazenados devem ser atualizados frequentemente para que as consultas possam ser úteis aos usuários. Dessa forma, é importante avaliar com qual frequência de tempo o usuário necessita que os dados sejam reprocessados e qual é o tempo mínimo para que isso ocorra. Para isso, é possível configurar tarefas por meio do Oozie para que os dados sejam processados com a frequência necessária. O que pode ser em questão de horas ou dias, dependendo da demanda de atualização e do tempo de processar os

dados. Em alguns casos, os usuários podem necessitar de dados em tempo real, ou próximo ao tempo real. Para esses casos é recomendável utilizar soluções de *streaming* como Spark Streaming, Storm, Flume ou uma combinação entre Flink e Kafka.

FINALIZANDO

Nesta aula, exploramos alguns temas complementares. Diversos outros temas poderiam ser explorados, uma vez que *big data* é uma área muito grande e possui diversas intersecções com várias outras áreas da computação.

No primeiro tema, vimos outros módulos que não foram abordados com tantos detalhes durante as outras aulas. Entre eles, as tecnologias de bancos SQL Impala e Accumulo, as plataformas de processamento e armazenamento em memória Redis e Ignite, o sistema de processamento e distribuição de dados através de grafos NiFi, e a interface web Ambari.

No segundo tema, discutimos as vantagens da abordagem de armazenamento de dados *data lake*, a necessidade de realizar uma governança dos dados através de seus metadados e os seus níveis de maturidade.

No terceiro tema, aprofundamos a discussão sobre os sistemas de recomendação. Discutimos os principais conceitos e motivações por trás dessa tecnologia e os principais tipos de recomendadores. Além disso, analisamos duas das principais abordagens de recomendação individualizada.

No quarto tema, conhecemos os principais conceitos a respeito da computação em nuvem (*cloud computing*), detalhamos os principais modelos de *cloud computing*, entendemos suas relações com a ideia de *big data as a service* e conhecemos as principais plataformas de *big data* oferecidas em *cloud computing.*

Por fim, no quinto tema, revisamos os detalhes que devem ser levados em consideração no design de um projeto de *big data*, bem como os principais requisitos que devem ser elencados.

REFERÊNCIAS

CHESSELL, N. et al. **Governing and managing Big Data for analytics and decision makers**. New York: IBM Redguides for Business Leaders, 2014.

DIXON, J. Pentaho, Hadoop, and Data Lakes. **James Dixon's Blog**, 2010. Disponível em: <https://jamesdixon.wordpress.com/2010/10/14/pentaho-hadoop-and-data-lakes/>. Acesso em: 26 nov. 2020.

GORELIK, A. **The enterprise Big Data lake:** delivering the promise of Big Data and Data Science. Sebastopol CA: O'Reilly Media inc., 2019.

LESKOVEC, J; RAJARAMAN, A; ULLMAN, J. **Mining of massive datasets**, Cambridge UK: Cambridge University Press, 2010.