SmarT: Uso de Aprendizado de Máquina para Filtragem e Recuperação Eficiente de Dados Espaciais e Temporais em Big Data

Sávio S. Teles de Oliveira[1], **Vagner J. do Sacramento Rodrigues**[1], **Wellington S. Martins**[1]

Instituto de Informática - Universidade Federal de Goiás (UFG)
Alameda Palmeiras, Quadra D, Câmpus Samambaia
131 - CEP 74001-970 - Goiânia - GO - Brazil

{savio.teles, vagner}@gogeo.io, wellington@inf.ufg.br

Abstract. *With the tremendous growth of big data time series, the efficient filtering and retrieval of large volumes of spatial and temporal data have become one of the biggest challenges for big data time series processing. Although some big data systems have been proposed to tackle these problems, none of them is considered a clear winner for all possible scenarios. This paper presents the SmarT search engine, a machine learning based solution that chooses the best big data system for filtering and retrieval of spatial and temporal data on the fly. In a detailed experimental evaluation, considering the Apache Spark, Elasticsearch, and SciDB big data systems, SmarT was able to reduce the response time in up to 22%.*

Resumo. *Com o aumento do volume de dados de séries temporais na era de Big Data, a filtragem e recuperação eficiente de um grande volume de dados, utilizados como entrada no processamento de séries temporais, são uns dos maiores desafios da área. Diversos sistemas de Big Data foram criados para lidar com estes desafios, mas nenhum possui o melhor desempenho de filtragem e recuperação de dados em todos os cenários com filtros espaciais e temporais. Este trabalho apresenta o motor de busca SmarT que utiliza algoritmos de aprendizado de máquina para escolher, em tempo real, o melhor sistema Big Data para filtrar e recuperar os dados de séries temporais. O trabalho avalia o Apache Spark, Elasticsearch e SciDB e mostra uma redução de quase 22% do tempo de resposta utilizando o SmarT.*

1. Introdução

Durante as últimas décadas, o processamento de séries temporais tem sido considerado um dos problemas mais desafiadores em mineração de dados [Fawaz et al. 2019]. Uma série temporal é uma coleção de observações feitas sequencialmente ao longo do tempo. Com o aumento da disponibilidade de dados na era de Big Data, a filtragem e recuperação eficiente de um grande volume de dados, utilizados como entrada no processamento de séries temporais, são uns dos maiores desafios da área [Wang et al. 2020]. Por isso, diversas soluções de plataformas Big Data foram propostas na literatura [Wang et al. 2020, Comber and Wulder 2019] utilizando sistemas de Big Data, tais como o Hadoop [1],

[1]https://hadoop.apache.org/

Spark [2], Elasticsearch [3] e SciDB [4], para indexação e gerenciamento dos dados de séries temporais.

Experimentos realizados [Zhang et al. 2018] mostram que nenhum sistema de Big Data tem desempenho melhor que os outros em todos os cenários com filtros espaciais e temporais. Fica a cargo do desenvolvedor da solução escolher previamente qual o sistema deseja utilizar para cada cenário de consulta. Essa escolha é uma tarefa complicada de ser feita dinamicamente, pois cada sistema Big Data tem características que permitem o melhor desempenho para determinados cenários de filtragem espacial e temporal. Além disso, os sistemas estão em constante evolução e em novas versões, podem surgir otimizações que se sobrepõem aos sistemas concorrentes.

Este trabalho apresenta um motor inteligente, denominado SmarT (smart Spatial-Temporal query engine), para filtragem e recuperação de dados espaço-temporais em Big Data utilizando algoritmos de aprendizado de máquina. O SmarT busca escolher, em tempo real, o melhor sistema de *Big Data* para processar a requisição, aplicando os filtros espaciais e temporais para recuperar os dados. O trabalho avaliou o Apache Spark, Elasticsearch e SciDB como sistemas de Big Data para filtragem e recuperação de dados de séries temporais, e comparou estes sistemas com o SmarT, tendo uma redução de quase 22% do tempo de resposta utilizando o SmarT.

O restante do trabalho está organizado como segue. A Seção 2 descreve as estratégias encontradas na literatura para processar grandes volumes de dados temporais. A Seção 3 apresenta o motor de busca SmarT e a Seção 4 descreve a metodologia e os resultados dos testes executados. A Seção 5 apresenta as conclusões deste trabalho e uma breve descrição dos trabalhos futuros.

2. Trabalhos Correlatos

Devido ao alto custo computacional e ao grande volume de dados disponíveis, diversos trabalhos em Big Data [Guo et al. 2014, Chi et al. 2016, Comber and Wulder 2019] desenvolveram soluções paralelas e distribuídas para processamento de séries temporais, particularmente para suportar a demanda por processamento em tempo real [Ma et al. 2015]. As pesquisas descrevem o uso de Big Data em diversos domínios de processamento de dados temporais e espaciais, tais como Smart Farming [Wolfert et al. 2017], monitoramento dos recursos hídricos [Wagner et al. 2014], análises de imagens de sensoriamento remoto [Rathore et al. 2015], IoT [Wang et al. 2015], sistemas de recomendação [Benabderrahmane et al. 2017] e mineração de dados de séries temporais [Fawaz et al. 2019].

Com o surgimento da computação nas nuvens, os usuários passaram a poder acessar os dados a qualquer momento em qualquer lugar para conduzir análises sobre as séries temporais. Em vista dessa facilidade, vários sistemas de computação distribuída foram construídos para o armazenamento e processamento de grande volume de dados [Almeer 2012]. O Google Earth Engine [Gorelick et al. 2017], por exemplo, é uma

[2]https://spark.apache.org
[3]https://www.elastic.co
[4]https://www.paradigm4.com/

das plataformas mais utilizadas no processamento de séries temporais de objetos espaciais, visando atender a uma grande gama de cientistas que não têm acesso a *clusters* de computadores.

Aliado a computação nas nuvens, o uso do modelo de programação MapReduce causou uma revolução no processamento e gerenciamento de dados de séries temporais [Guo et al. 2017]. O MapReduce facilitou a construção de trabalhos para mineração de dados de séries temporais de imagens de satélite [Lin et al. 2013, de Assis et al. 2017], de objetos espaciais e temporais [Song et al. 2015, Patterson 2011] e objetos 3D [Van Den Bergh et al. 2012].

O aumento de demanda por consumir os dados das séries temporais do lado da aplicação, levou ao surgimento dos bancos de dados vetoriais multidimensionais para o armazenamento, gerenciamento, processamento e análise das séries temporais, tais como o SciDB [Brown 2010] e o Rasdaman [Baumann et al. 1997]. Estes bancos de dados têm sido utilizados em diversos trabalhos [Lu et al. 2016, Camara et al. 2016] para processamento eficiente de dados multidimensionais, como séries temporais de objetos espaciais.

Os trabalhos encontrados na literatura utilizam os mais diversos sistemas de Big Data para processamento de séries temporais. Experimentos realizados [Zhang et al. 2018] mostram que nenhum sistema de Big Data tem desempenho melhor que os outros em todos os cenários de filtros espaciais e temporais. Entretanto, não foi encontrado nenhum trabalho na literatura que fosse capaz de explorar os diversos sistemas de Big Data disponíveis, de acordo com o cenário de consulta, para filtragem e recuperação dos dados de séries temporais.

3. SmarT: Motor de Busca Inteligente para Filtragem e Recuperação de Dados Espaciais e Temporais

O SmarT é um motor inteligente que utiliza algoritmos de aprendizado de máquina na filtragem e recuperação de dados de séries temporais, de forma eficiente, explorando os diversos sistemas de Big Data disponíveis. O Smart faz a predição do tempo de execução, utilizando um algoritmo de aprendizado de máquina supervisionado, para cada sistema Big Data e escolhe aquele de menor tempo de resposta estimado, tendo como entrada: i) as restrições espaciais e temporais; ii) as métricas do *cluster* obtidas no sistema de monitoramento e iii) a quantidade aproximada de resultados obtida no sistema de indexação da plataforma. A Figura 1 apresenta a arquitetura do SmarT.

As restrições espaciais e temporais são enviadas pelo cliente para a API do SmarT e os dados também são retornados pela API. O cliente da API pode ser, por exemplo, retornar dados das séries temporais entre os anos de 2010 e 2020 (restrição temporal) da região Amazônica (restrição espacial). Se nenhuma restrição espacial ou temporal for informada, a API retorna toda a base de dados existente. Para evitar o estouro de memória e aumentar a eficiência de recuperação, o resultado é retornado em lotes de dados, cujo tamanho é configurado na requisição à API.

O sistema de monitoramento coleta periodicamente métricas dos servidores do *cluster*, tais como uso da rede, área de swap, memória e CPU. O SmarT utiliza o sistema de monitoramento do Ambari [5], que permite a coleta de métricas dos servidores e de

[5]https://ambari.apache.org/

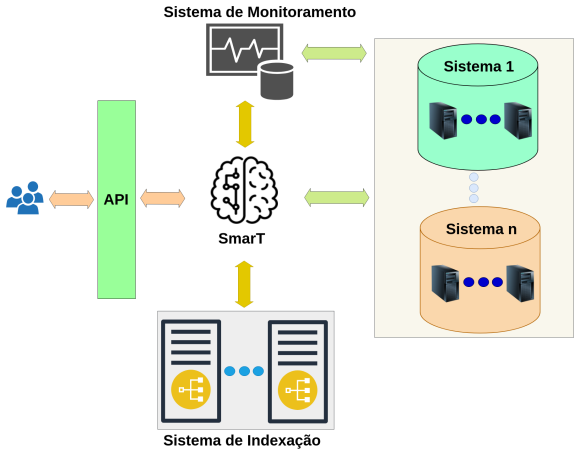

Figura 1. Arquitetura do SmarT.

vários sistemas de Big Data, como Hadoop e Spark. A arquitetura é extensível, de forma que, é possível utilizar outros sistemas de monitoramento para coletar as métricas do *cluster*. Estas métricas também são utilizadas como entrada para o SmarT.

O SmarT assume que os dados das séries temporais já estão armazenados e indexados em cada sistema Big Data. Nosso trabalho utiliza a plataforma Big Data descrita em [Oliveira 2019], que replica os dados das séries temporais no sistema de armazenamento e indexação de cada sistema Big Data. Experimentos nesta plataforma mostraram que ela é capaz de ingerir e indexar mais de 6 milhões de observações de séries temporais por segundo [Oliveira 2019]. O SmarT também assume que os dados das séries temporais estão indexados em um Sistema de Indexação de baixa latência de consulta, que permite obter a quantidade aproximada de resultados dos filtros espaciais e temporais, que é uma das entradas do algoritmo de aprendizado de máquina do SmarT. Nosso trabalho utiliza o Sistema de Indexação de [Oliveira 2019] que permite indexar e retornar a quantidade aproximada de resultados de forma eficiente. A plataforma Big Data descrita em [Oliveira 2019] pode ser substituída por outra que armazene e indexe os dados nos sistemas de Big Data e que possua um Sistema de Indexação de baixa latência.

O algoritmo de aprendizado de máquina utilizado pelo SmarT (apresentado na Seção 3.1) tem como entrada os filtros de consulta, métricas de monitoramento, a quantidade aproximada de resultados e o tempo de resposta de cada sistema Big Data para o determinado filtro. Os dados de treinamento do SmarT são obtidos realizando um conjunto de consultas pré-definidas sobre cada um dos sistemas Big Data existentes, filtrando e recuperando os dados das séries temporais, e coletando os dados de entrada para o treinamento. O treinamento do algoritmo é disparado manualmente utilizando estes dados e, ao final do treinamento, o modelo gerado é atualizado no SmarT, que passa a utilizá-lo na predição do tempo de resposta de cada sistema de Big Data.

3.1. Uso do Aprendizado de Máquina na Seleção em Tempo Real do Sistema de *Big Data* para Filtragem e Recuperação de Dados de Séries Temporais

O SmarT busca minimizar o tempo de resposta da filtragem temporal e espacial dos dados das séries temporais, escolhendo o sistema de Big Data com menor tempo de resposta previsto pelo algoritmo de aprendizado de máquina. Dado que o tempo de resposta é uma

variável contínua, o problema de minimizá-lo foi mapeado com uma modelagem preditiva de regressão com o objetivo de aproximar a função de mapeamento das características de entrada para a variável de saída contínua que representa o tempo de resposta. Utilizando a modelagem preditiva de regressão, mesmo que não seja escolhido o melhor sistema, aumentamos a possibilidade de se escolher um dos melhores.

Na nossa proposta de modelagem preditiva de regressão, o tempo de resposta da consulta de cada sistema de *Big Data* é a variável de resposta (dependente) e as variáveis explicativas (independentes) são compostas pelas restrições da consulta, métricas de monitoramento do *cluster* e o número estimado de resultados na consulta. Para cada sistema de *Big Data* é construído um modelo dentro do SmarT, onde a variável dependente representa o tempo de resposta do sistema de *Big Data*. Denotaremos a variável de resposta por y e o conjunto de variáveis explicativas por $x_1, x_2, ..., x_p$, onde p indica o número de variáveis explicativas. O relacionamento entre y e $x_1, x_2, ..., x_p$ é aproximado pelo modelo de regressão da equação 1:

$$y = f(x_1, x_2, ..., x_p) + \varepsilon \tag{1}$$

onde ε é adotado como um erro randômico representando a discrepância na aproximação, para levar em conta os possíveis erros de predição do modelo. Assim, a função $f(x_1, x_2, ..., x_p)$ descreve o relacionamento entre y e $x_1, x_2, ..., x_p$.

Existem diversos modelos de regressão [Chatterjee and Hadi 2015] que podem ser utilizados para estimar o tempo de resposta de cada sistema. Nossa solução utiliza o método *ensemble* XGBoost (Extreme Gradient Boosting) [Chen and Guestrin 2016], escolhido pelo seu destaque em várias competições de ciência de dados com dados tabulares de entrada (que é exatamente nosso formato de entrada) [Nielsen 2016]. O XGBoost é um algoritmo supervisionado que implementa um processo chamado *boosting* para produzir modelos com maior acurácia. A entrada para o algoritmo são pares de exemplos de treinamento ($\vec{x}_1$, y_1), ($\vec{x}_2$, y_2), ($\vec{x}_p$, y_p), onde $\vec{x}$ é um vetor de características representando as variáveis explicativas e y é o tempo de resposta da consulta. Sabendo que o tempo de resposta (*response_time*) representa a variável de resposta y, então para cada par de entrada do treinamento ($\vec{x}_i$, y_i), a variável de resposta y_i é definida pela equação 2 e o vetor de características $\vec{x}_i$ pela equação 3:

$$y_i = response\_time_i \tag{2}$$

$$\begin{aligned} \vec{x}_i = (&area_i, time\_interval_i, count_i, swap\_free_i, mem\_free_i, \\ &load\_one_i, load\_five_i, load\_fifteen_i, \\ &cpu\_user_i, cpu\_system_i, bytes\_in_i, bytes\_out_i) \end{aligned} \tag{3}$$

Os valores das variáveis explicativas *area* e *time_interval* são obtidos a partir dos filtros espacial e temporal e a variável *count* é gerada a partir do Sistema de Indexação, que retorna o número estimado de resultados. Combinando as variáveis *area*, *time_interval* e *count* é possível estimar, de forma mais acurada, o custo de recuperação dos dados de resposta da consulta. As demais métricas são obtidas do sistema de monitoramento do *cluster* e são importantes porque impactam na performance de um sistema em um ambiente distribuído. São coletadas a média das métricas de todos os servidores, para

representar o estado do *cluster* no momento da consulta. Diversas outras variáveis podem ser acopladas ao modelo de aprendizado de máquina do SmarT, mas nosso trabalho definiu um escopo com as variáveis apresentadas acima. O SmarT permite também que outros métodos de aprendizado possam ser utilizados, substituindo o XGBoost atual.

4. Avaliação de Desempenho

Esta seção avalia a capacidade do SmarT de escolher o melhor sistema de Big Data para cada cenário. Três sistemas são avaliados neste contexto: i) Elasticsearch versão 7.0.1, ii) SciDB versão 18.3 e iii) Apache Spark versão 2.3.2. Estes sistemas já estão integrados a plataforma Big Data de [Oliveira 2019] utilizada no nosso trabalho. A hipótese deste trabalho é que nenhum destes três sistemas consegue ser eficiente em todos os cenários de consultas com filtros espaciais e temporais sobre a base de dados.

4.1. Ambiente Experimental e Bases de Dados

Os testes foram realizados em um *cluster* de computadores com 8 servidores no ambiente de *Cloud Computing* da Amazon EC2[6]. Cada um destes servidores está equipado com processadores Intel Xeon® Platinum 8175 de até 3,1 GHz com 16 vCPUs, 128 GB de memória RAM e 600 GB de SSD. As máquinas foram conectadas por uma rede Ethernet 10 Gbit/segundo com largura de banda dedicada de 3500 Mbps. Foram armazenadas mais de 7,2 bilhões de observações de dados de séries temporais de imagens de satélite de uma área geográfica de 800.824 km^2 que cobre um pedaço do Centro-Oeste e Sudeste do Brasil. As séries temporais são referentes aos anos entre 2000 e 2019 e foram extraídas do produto MOD13Q1 da NASA[7] (National Aeronautics and Space Administration), com uma periodicidade de 16 dias entre cada observação e resolução espacial de 250 metros.

Foram geradas 1983 restrições espaciais e temporais a partir de: i) 247 filtros espaciais extraídos da área do estado de Goiás no Brasil, ii) 7 filtros temporais no intervalo de 2016 a 2019 e iii) 1729 (247 * 7) filtros espaço-temporais que combinam os 247 filtros espaciais e 7 temporais. Utilizando os filtros espaciais e temporais, as consultas retornam de 0 a 2 bilhões de resultados. Cada consulta foi executada 10 vezes em cada um dos três sistemas Big Data, coletando as métricas que compõem o vetor de características $\vec{x}_i$ e o tempo de execução da consulta (variável de resposta y_i). Dos dados coletados, 75% foram utilizadas para treinar o SmarT e o restante para avaliar a acurácia do SmarT em escolher o melhor sistema Big Data para a consulta. Na implementação do SmarT, foi utilizado o algoritmo supervisionado XGBoost [Chen and Guestrin 2016] da plataforma *open source* H2O [8], versão 3.24.0.4. Os experimentos foram executados utilizando o método de validação cruzada denominado k-fold, com k=5, com os valores padrões para os parâmetros do algoritmo XGBoost na plataforma H2O.

O SciDB e o Apache Spark foram instalados com as configurações padrões, sem nenhuma otimização ou customização. Em relação ao Elasticsearch, foi alterado o valor padrão de 1 GB para memória *heap* da JVM[9] para 48 GB, já que as consultas estavam causando estouro da memória *heap* durante os testes. O restante da memória da máquina

[6]https://aws.amazon.com/pt/ec2/.

[7]Esta base de dados foi disponibilizada pelo LAPIG (Laboratório de Processamento de Imagens e Geoprocessamento) https://www.lapig.iesa.ufg.br/lapig

[8]https://www.h2o.ai/

[9]Elasticsearch foi desenvolvido na linguagem Java

ficou alocada para os outros sistemas instalados nos servidores. Os testes no Spark foram executados utilizando o ecossistema Hadoop versão 3.1.1, com o Apache Yarn escalonando as consultas e gerenciando os recursos do *cluster* consumidos pelo Spark, com os dados armazenados no HDFS no formato Parquet com a compressão Snappy [10].

4.2. Resultados

Esta seção apresenta os resultados da avaliação do SmarT. A Tabela 1 mostra uma comparação da porcentagem de vezes que cada um dos três sistemas foi a melhor escolha utilizando três faixas de número de resultados retornados: i) até 100.000 resultados, ii) entre 100.000 e 500.000 e iii) mais de 500.000 resultados. Para um número de resultados pequeno (menos de 100 mil), o Elasticsearch é a melhor opção entre os três sistemas, mas a partir de 100 mil resultados retornados, o SciDB e o Spark se tornam melhores opções. O Elasticsearch possui um mecanismo interno de indexação muito eficiente, mas sua performance depende do tamanho do conjunto de dados retornados. Quando o resultado a ser recuperado fica maior que um determinado limite[11], o tempo de resposta aumenta consideravelmente [Thacker et al. 2016], de forma que, o Elasticsearch passa a ter um desempenho pior que os outros dois sistemas.

	Até 100 mil	Entre 100 e 500 mil	Mais que 500 mil
Elasticsearch	90%	0%	0%
SciDB	1%	22%	64%
Apache Spark	9%	78%	36%

Tabela 1. Comparação da porcentagem de vezes que cada um dos três sistemas foi a melhor escolha utilizando três faixas para avaliação da recuperação dos resultados: i) até 100 mil resultados, ii) entre 100 e 500 mil resultados e iii) mais de 500 mil resultados.

O Spark armazena os dados no formato Parquet, que consome menos espaço em disco do que o formato JSON [Alonso Isla 2018] utilizado pelo Elasticsearch e, por isso, consegue ter melhor desempenho para um número de resultados maior que 100 mil. Mas, para um resultado com mais de 500 mil tuplas, o Spark tem um desempenho pior que o SciDB porque tem que percorrer todos os arquivos Parquet para filtrar os resultados, e quando a base de dados é maior que o tamanho da memória, os resultados intermediários precisam ser serializados como objetos Java e armazenados em dispositivos de armazenamento secundários (disco ou SSD) ou recalculados. Este *overhead* é crítico para o Spark e impacta negativamente no seu desempenho.

O SciDB possui um sistema interno eficiente de recuperação dos dados do disco que fica evidente com resultados maiores que 500 mil, onde seu desempenho é melhor que o Spark. O SciDB é projetado para lidar com dados de séries temporais, ao contrário do Spark, que é um motor de processamento geral. Este cenário fica claro para um grande volume de dados de séries temporais, onde o SciDB explora seu eficiente sistema de indexação e recuperação de dados multidimensionais [Zhang et al. 2016].

Para efeitos de comparação dos três sistemas, o Elasticsearch é recomendado para um volume de resultados menor, o Spark para um volume de resultados médio e o SciDB

[10] https://github.com/google/snappy

[11] Este limite depende muito da configuração do Elasticsearch e da infraestrutura de *hardware* do **cluster**.

para um volume maior. O desafio aqui, entretanto, é a escolha do sistema, já que a definição de um conjunto de resultados menor, médio ou maior é subjetiva e depende da configuração de *hardware* do *cluster*, dos sistemas disponíveis e dos cenários de filtragem espacial e temporal destes sistemas. É possível perceber na Tabela 1, que nenhum sistema domina totalmente os outros, analisando a porcentagem de vezes que cada um teve o menor tempo de resposta em cada faixa de dados retornados na consulta. Este resultado leva a elaboração de uma hipótese que outras variáveis também impactam no tempo de resposta dos sistemas.

O algoritmo de aprendizado de máquina utiliza as seguintes variáveis no momento da consulta: *area*, *time_interval*, *count*, *swap_free*, *mem_free*, *load_one*, *load_five*, *load_fifteen*, *cpu_user*, *cpu_system*, *bytes_in*, *bytes_out*. O SmarT obteve uma acurácia geral de 90,1% na escolha do melhor sistema e, mesmo nos casos em que não escolhe o melhor, existe uma diferença pequena de tempo de resposta se tivesse escolhido sempre o melhor sistema para cada consulta. Em 95% dos casos, o sistema escolhido pelo SmarT é o melhor ou tem tempo de resposta até 10% maior do que o tempo do melhor sistema.

Como pode ser visto na Tabela 2, utilizando o SmarT, o tempo médio de resposta foi de 496ms contra 482ms se tivesse escolhido sempre o melhor sistema (coluna "Best"), ou seja, se o SmarT nunca errasse na escolha. Os tempos apresentados na Tabela 2 foram calculados a partir da média das execuções de todas as consultas. O tempo do SmarT é calculado a partir dos tempos de resposta do sistema Big Data escolhido em cada consulta. Para cada consulta, é adicionado no tempo final do SmarT, o tempo para a coletar as métricas do *cluster* e consultar o sistema de indexação para estimar o número de resultados da consultas. A Tabela 2 apresenta os resultados do SmarT com este *overhead* (em média 3ms) incluso.

	SmarT	Best	Apache Spark	Elasticsearch	SciDB
Todos os Sistemas	496	482	822	6291	605
Elasticsearch + SciDB	537	533	∅	6291	605
Elasticsearch + Spark	800	796	822	6291	∅
Spark + SciDB	527	513	822	∅	605

Tabela 2. Comparação do tempo médio de resposta (ms) entre o SmarT e os três sistemas. A coluna "Best" indica o tempo médio de resposta se fosse escolhida sempre a melhor opção.

A Tabela 2 também apresenta a comparação do tempo médio de resposta do SmarT em relação ao Spark, Elasticsearch e SciDB. Nesta tabela, também é apresentada uma comparação do tempo de resposta se o SmarT escolher todos os sistemas ou somente dois deles. O SciDB teve o menor tempo de resposta seguido pelo Spark e Elasticsearch, respectivamente. De forma geral, o SmarT teve um tempo médio de resposta próximo ao "Best" com diferença de menos de 3% para todas as combinações de sistemas. Com dois sistemas, a dupla Spark e SciDB teve o melhor desempenho, seguida pela dupla Elasticsearch e SciDB com uma pequena diferença de tempo médio de 10 ms entre as duas. A dupla Elasticsearch e Spark teve o pior desempenho, sendo 33% pior que as outras opções. O SciDB é o sistema mais importante dentre os três para reduzir o tempo de resposta. O tempo médio de todas as consultas no SciDB é o mais baixo entre as três porque possui o melhor desempenho quando a consulta retorna um alto número de

resultados. Para estes casos, os tempos no Elasticsearch e Spark tem o pior desempenho, como pode ser visto na Tabela 1, o que impacta o tempo médio final.

4.3. Discussão dos Resultados

Esta seção apresenta a discussão dos resultados da avaliação do SmarT. Na estimativa do tempo de resposta dos sistemas utilizando o XGBoost, a característica mais importante na tomada de decisão do SmarT foi a variável *count*, como já era esperado pelos resultados apresentados na Tabela 1. Na estimativa do tempo do Elasticsearch, outras características tiveram impacto, sendo elas por ordem de prioridade: *area*, *time_interval* e *bytes_out*. As variáveis *area* e *time_interval* são ligadas ao número de resultados retornados, já que, geralmente, quanto maior for a restrição espacial e temporal maior será o número de resultados retornados. A característica *bytes_out* teve impacto no Elasticsearch devido ao custo de tráfego dos resultados da consulta, inerente ao Elasticsearch que não possui otimizações neste sentido, e a situação da rede no momento da consulta.

Na estimativa do tempo do Spark, além das características *count*, *area* e *time_interval*, a quantidade de memória livre teve impacto na previsão. Isto acontece porque o Spark faz o processamento das consultas utilizando computação em memória e, por isso, é impactado pela quantidade de memória livre no momento da consulta, tendo, em alguns casos, de usar a área de *swap* do sistema operacional quando não existe memória suficiente. Na estimativa de tempo do SciDB, além das características *count*, *area* e *time_interval*, as variáveis *load_five*, *load_five* e *cpu_system* tiveram impacto na previsão do tempo de resposta. O SciDB possui um mecanismo interno de indexação e armazenamento otimizado para recuperação de dados do disco, mas com alto custo de processamento. Por isso, a situação do *cluster* em relação ao uso de processamento nas máquinas impacta no tempo de resposta do SciDB.

O recomendado é ter os três sistemas para ter o melhor desempenho possível, mas caso não seja possível, analisando os resultados das Tabelas 2 e 1, a dupla Elasticsearch e SciDB deve ser utilizada se houver um cenário em que há várias consultas com restrições que retornam poucos resultados. Quando não se sabe o cenário das consultas dos clientes, em relação as restrições de filtragem espacial e temporal, ou em cenários com restrições que retornam muitos resultados, o ideal é utilizar a dupla Spark e SciDB.

Na escolha de um sistema, o SciDB provou ter melhor desempenho que o Spark e o Elasticsearch, mas utilizar o SmarT com os três sistemas é a melhor opção. O SciDB teve um tempo médio de resposta 21,9% maior que o SmarT quando todos os sistemas estão disponíveis. Mesmo em relação as duplas Elasticsearch e SciDB e Spark e SciDB com o SmarT, o SciDB teve tempo médio de resposta 12,6% e 14,8% maior do que as duplas, respectivamente. Assim, o SmarT, com acurácia de 90,1% e redução de quase 22% em relação ao SciDB no tempo de resposta, é uma boa opção quando é possível ter mais de um sistema de Big Data no *cluster*.

5. Conclusões

O grande volume de dados não estruturados, com alta dimensionalidade, sendo gerados em grande quantidade a todo momento coloca o problema de processamento de séries temporais no contexto de Big Data [Fan et al. 2014]. A filtragem e recuperação eficiente de um grande volume de dados, utilizados como entrada no processamento de séries

temporais, são uns dos maiores desafios abertos na área. Este trabalho propõe um motor inteligente, denominado SmarT, capaz de filtrar e recuperar os dados das séries temporais de forma eficiente. O objetivo do SmarT é utilizar um algoritmo de aprendizado de máquina para escolher o melhor sistema Big Data, entre vários existentes, para filtrar e recuperar os dados da consulta no menor tempo possível.

O SmarT teve uma acurácia de mais de 90% na escolha do melhor sistema entre o Apache Spark, Elasticsearch e SciDB. Com isso houve uma redução de 22% no tempo de resposta utilizando o SmarT do que se fosse utilizado apenas um dos três sistemas. Seria praticamente inviável realizar essa escolha de forma manual, em cada consulta, devido ao número de combinações possíveis das variáveis utilizadas pelo SmarT. O uso do SmarT permite que os pesquisadores se concentrem na construção de algoritmos de processamento de séries temporais, sem se preocupar com o desafio da filtragem e recuperação eficiente do grande volume de dados.

O treinamento do SmarT é realizado manualmente executando um conjunto de consultas sobre os dados de treinamento dos algoritmos de aprendizado de máquina. Para gerar um conjunto de treinamento que represente melhor os cenários de consultas feitas pelos usuários, iremos modificar o SmarT, em trabalhos futuros, para que ele possa gerar o conjunto de treinamento a partir de consultas reais dos usuários, treinando o modelo em períodos de ociosidade do *cluster*, de forma automática. Além disso, iremos adicionar métricas específicas de cada sistema, como o tamanho da fila de espera por execução em cada sistema.

Agradecimentos

Este trabalho foi financiado pela Companhia Paranaense de Energia (Copel), no projeto de P&D ANEEL-Copel Distribuição número 2866-04842017.

Referências

Almeer, M. H. (2012). Cloud hadoop map reduce for remote sensing image analysis. *Journal of Emerging Trends in Computing and Information Sciences*, 3(4):637–644.

Alonso Isla, Á. (2018). Hdfs file formats: Study and performance comparison. Master's thesis, Universidad de Valladolid. Escuela Técnica Superior de Ingenieros de Telecomunicación.

Baumann, P., Furtado, P., Ritsch, R., and Widmann, N. (1997). The rasdaman approach to multidimensional database management. In *Symposium on Applied Computing: Proceedings of the 1997 ACM symposium on Applied computing*, volume 1997, pages 166–173.

Benabderrahmane, S., Mellouli, N., Lamolle, M., and Paroubek, P. (2017). Smart4job: A big data framework for intelligent job offers broadcasting using time series forecasting and semantic classification. *Big Data Research*, 7:16–30.

Brown, P. G. (2010). Overview of scidb: large scale array storage, processing and analysis. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*, pages 963–968. ACM.

Camara, G., Assis, L. F., Ribeiro, G., Ferreira, K. R., Llapa, E., and Vinhas, L. (2016). Big earth observation data analytics: matching requirements to system architectures.

In *Proceedings of the 5th ACM SIGSPATIAL International Workshop on Analytics for Big Geospatial Data*, pages 1–6. ACM.

Chatterjee, S. and Hadi, A. S. (2015). *Regression analysis by example*. John Wiley & Sons.

Chen, T. and Guestrin, C. (2016). Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining*, pages 785–794. ACM.

Chi, M., Plaza, A., Benediktsson, J. A., Sun, Z., Shen, J., and Zhu, Y. (2016). Big data for remote sensing: Challenges and opportunities. *Proceedings of the IEEE*, 104(11):2207–2219.

Comber, A. and Wulder, M. (2019). Considering spatiotemporal processes in big data analysis: Insights from remote sensing of land cover and land use. *Transactions in GIS*.

de Assis, L. F. F. G., de Queiroz, G. R., Ferreira, K. R., Vinhas, L., Llapa, E., Sanchez, A. I., Maus, V., and Câmara, G. (2017). Big data streaming for remote sensing time series analytics using mapreduce. *Revista Brasileira de Cartografia*, 69(5).

Fan, J., Han, F., and Liu, H. (2014). Challenges of big data analysis. *National science review*, 1(2):293–314.

Fawaz, H. I., Forestier, G., Weber, J., Idoumghar, L., and Muller, P.-A. (2019). Deep learning for time series classification: a review. *Data Mining and Knowledge Discovery*, 33(4):917–963.

Gorelick, N., Hancher, M., Dixon, M., Ilyushchenko, S., Thau, D., and Moore, R. (2017). Google earth engine: Planetary-scale geospatial analysis for everyone. *Remote Sensing of Environment*, 202:18–27.

Guo, H., Liu, Z., Jiang, H., Wang, C., Liu, J., and Liang, D. (2017). Big earth data: a new challenge and opportunity for digital earth's development. *International Journal of Digital Earth*, 10(1):1–12.

Guo, H., Wang, L., Chen, F., and Liang, D. (2014). Scientific big data and digital earth. *Chinese science bulletin*, 59(35):5066–5073.

Lin, F.-C., Chung, L.-K., Wang, C.-J., Ku, W.-Y., and Chou, T.-Y. (2013). Storage and processing of massive remote sensing images using a novel cloud computing platform. *GIScience & Remote Sensing*, 50(3):322–336.

Lu, M., Pebesma, E., Sanchez, A., and Verbesselt, J. (2016). Spatio-temporal change detection from multidimensional arrays: Detecting deforestation from modis time series. *ISPRS Journal of Photogrammetry and Remote Sensing*, 117:227–236.

Ma, Y., Wu, H., Wang, L., Huang, B., Ranjan, R., Zomaya, A., and Jie, W. (2015). Remote sensing big data computing: Challenges and opportunities. *Future Generation Computer Systems*, 51:47–60.

Nielsen, D. (2016). Tree boosting with xgboost-why does xgboost win"every"machine learning competition? Master's thesis, NTNU.

Oliveira, S. S. T. d. (2019). *Explorando paralelismo em big data no processamento de séries temporais de imagens de sensoriamento remoto*. PhD thesis, Universidade Federal de Goiás. Instituto de Informática.

Patterson, J. (2011). Lumberyard: Time series indexing at scale. In *OSCON 2011 - O'Reilly Conferences*, Portland, USA.

Rathore, M. M. U., Paul, A., Ahmad, A., Chen, B.-W., Huang, B., and Ji, W. (2015). Real-time big data analytical architecture for remote sensing application. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 8(10):4610–4621.

Song, W., Jin, B., Li, S., Wei, X., Li, D., and Hu, F. (2015). Building spatiotemporal cloud platform for supporting gis application. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 1:55–62.

Thacker, U., Pandey, M., and Rautaray, S. S. (2016). Performance of elasticsearch in cloud environment with ngram and non-ngram indexing. In *2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT)*, pages 3624–3628. IEEE.

Van Den Bergh, F., Wessels, K. J., Miteff, S., Van Zyl, T. L., Gazendam, A. D., and Bachoo, A. K. (2012). Hitempo: a platform for time-series analysis of remote-sensing satellite data in a high-performance computing environment. *International journal of remote sensing*, 33(15):4720–4740.

Wagner, W., Fröhlich, J., Wotawa, G., Stowasser, R., Staudinger, M., Hoffmann, C., Walli, A., Federspiel, C., Aspetsberger, M., Atzberger, C., et al. (2014). Addressing grand challenges in earth observation science: The earth observation data centre for water resources monitoring. *ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences*, 2(7).

Wang, F., Li, M., Mei, Y., and Li, W. (2020). Time series data mining: A case study with big data analytics approach. *IEEE Access*, 8:14322–14328.

Wang, Y., Yuan, J., Chen, X., and Bao, J. (2015). Smart grid time series big data processing system. In *Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), 2015 IEEE*, pages 393–400. IEEE.

Wolfert, S., Ge, L., Verdouw, C., and Bogaardt, M.-J. (2017). Big data in smart farming-a review. *Agricultural Systems*, 153:69–80.

Zhang, X., Khanal, U., Zhao, X., and Ficklin, S. (2016). Understanding software platforms for in-memory scientific data analysis: A case study of the spark system. In *2016 IEEE 22nd International Conference on Parallel and Distributed Systems (ICPADS)*, pages 1135–1144. IEEE.

Zhang, X., Khanal, U., Zhao, X., and Ficklin, S. (2018). Making sense of performance in in-memory computing frameworks for scientific data analysis: A case study of the spark system. *Journal of Parallel and Distributed Computing*, 120:369–382.