DStream: Plataforma de coleta, processamento e armazenamento de streams de dados de sensoriamento remoto

Sávio Oliveira[1], Vagner Rodrigues[1], Laerte Ferreira[2], Claudinei Santos[2], Wellington Martins[1]

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

[2]LAPIG - CAMPUS II Samambaia - Cx. POSTAL 131
CEP: 74001-970 - Goiânia - GO - Brasil

{savio.teles,vagner}@gogeo.io,{lapig.ufg,claudineisantosnx}@gmail.com,
wellington@inf.ufg.br

Abstract. *Currently, one of the best applications has been in identifying transformations on Earth surface, which is changing at an unprecedented rate. Hence, great efforts have been made to create solutions to identify these changes by processing large volumes of data (Big Data) continuously generated by several sources (streaming). The goal of this paper is to explore the parallel architectures to proposes a new distributed solution for the collect, storing, indexing and processing of time series of remote sensing images.*

Resumo. *Atualmente, uma das maiores aplicações de séries temporais tem sido na identificação de transformações da superfície do planeta Terra, que está mudando a uma taxa sem precedentes. Por isso, grandes esforços tem sido feitos para criar soluções para identificar estas mudanças processando grandes volumes de dados (Big Data) gerados continuamente por diversas fontes (streaming). O objetivo deste trabalho é explorar as arquiteturas paralelas para propor uma nova solução distribuída para coleta, armazenamento, indexação e processamento de séries temporais de imagens de sensoriamento remoto.*

1. Introdução

A superfície do planeta Terra está mudando a uma taxa sem precedentes, no qual ecossistemas de florestas diminuem a uma velocidade alarmante e áreas urbanas e agrícolas expandem em volta do espaço natural. Análise das séries temporais de imagens de sensoriamento remoto tem se tornado indispensáveis na identificação destas mudanças.

Na classe de problemas computacionalmente custosos, a análise de séries temporais é um dos problemas com maior demanda de poder computacional [Rakthanmanon et al. 2013] e podem se beneficiar das arquiteturas paralelas. O grande volume de dados, com alta dimensionalidade, sendo gerados em grande quantidade a todo momento traz desafios únicos para explorar análise de séries temporais no cenário de *Big Data* [Fan et al. 2014]. Mais especificamente na área de sensoriamento remoto, podemos dizer que estamos vivendo uma era do *Big Remote Sensing Data* [Chi et al. 2016].

As soluções paralelas centralizadas (MultiCore e ManyCore), com memória compartilhada, não são capazes de processar séries temporais no contexto *Big Data*

[Sottile et al. 2009] e, por isso, diversas soluções paralelas com memória distribuída tem sido propostas para processar esse grande volume de dados de séries temporais [Chen et al. 2014]. As melhores soluções paralelas para *Big Data* utilizam ambas arquiteturas de memórias (compartilhada e distribuída) provendo flexibilidade para otimizar o paralelismo em aplicações para extrair o máximo de eficiência [Sottile et al. 2009].

No contexto de *Big Remote Sensing Data*, não foram encontrados algoritmos eficientes para processamento de streams de dados de séries temporais de imagens de sensoriamento remoto, sendo um dos grandes desafios e oportunidades da área [Ma et al. 2015]. Foi proposta a plataforma DistSensing [de Oliveira et al. 2017] para processamento de séries temporais, mas que não permite trabalhar com *streams* de dados. O objetivo deste trabalho é propor uma plataforma, denominada DStream, para coletar, armazenar e processar os dados de *Big Remote Sensing Data* que chegam a todo momento de diversas fontes. Na Seção 2 apresentamos a descrição desta plataforma, com a avaliação sendo apresentada na Seção 3 e as conclusões na Seção 4.

2. DStream

O processamento de *streams* de dados deve ser realizado no momento em que os dados chegam ao sistema com uma baixa latência de resposta. Para atender a estes requisitos, este trabalho apresenta uma solução, denominada DStream, para o processamento de *streams* de dados de séries temporais de imagens de sensoriamento remoto. A arquitetura dessa solução, apresentada na Figura 1, é constituída por módulos de coleta, transformação e armazenamento dos dados de séries temporais. Os dados são coletados de diversas fontes que sejam capazes de enviar dados históricos de imagens de sensoriamento remoto.

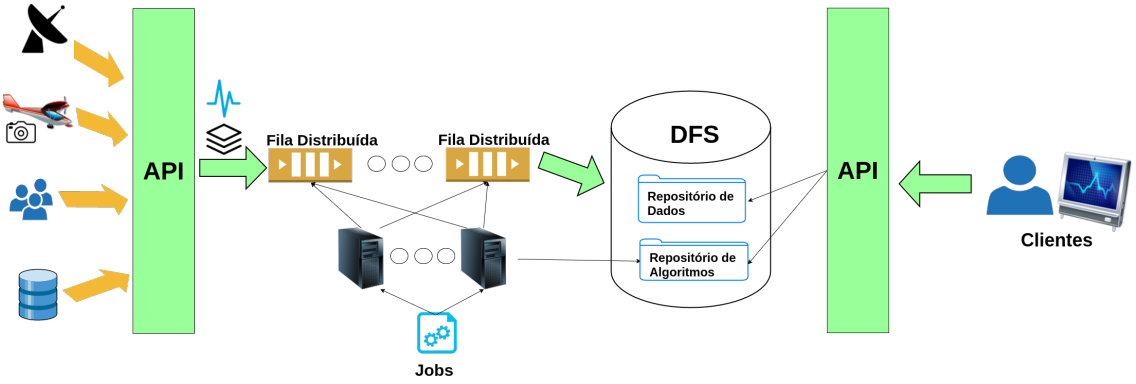

Figura 1. Arquitetura do Sistema de Processamento de *Stream* de Dados de Séries Temporais

A fila de entrada é distribuída e visível a todos os servidores do *cluster*, em que qualquer servidor é capaz de recuperar um objeto da fila e processar. Este objeto recuperado contém um bloco com um conjunto de *pixels* e cada *pixel* carrega a série histórica dos seus valores de NDVI. Esse bloco é processado utilizando os algoritmos de análise de séries temporais armazenados no sistema de arquivos distribuído (DFS). Esses algoritmos são implementados por um especialista em alguma área, que deseja realizar análises sobre as séries temporais de imagens de sensoriamento remoto. Para isso, o cliente deve submeter um código para a API da DStream, que armazena este código no DFS para ser então

utilizado no processamento dos blocos de *pixels*. Esses códigos devem seguir um padrão, de receber como entrada a série temporal de um *pixel* e gerar uma saída no formato texto.

Cada *job* no sistema é executado por vários servidores do *cluster*, cabendo a plataforma garantir a tolerância a falhas, o controle de fluxo e o balanceamento dos *jobs* entre os servidores. Cada objeto enviado na fila distribuída é recuperado por um *job* executado em uma máquina. Esse *job* é responsável por recuperar cada algoritmo de análise de séries temporais de imagens de sensoriamento remoto armazenados no DFS e executar estes algoritmos sobre o objeto contendo um bloco de séries temporais de vários *pixels*. Para cada série temporal de um *pixel*, os algoritmos são executados e a saída enviada para a fila de saída do sistema, contendo o identificador do *pixel* e do algoritmo como chave e o texto retornado por cada algoritmo como valor.

Esta arquitetura, na qual objetos são recuperados de uma fila de entrada, processados e enviados para uma fila de saída, permite que outros *jobs* sejam integrados facilmente ao sistema, bastando adicionar filas distribuídas e *jobs* para consumir dados destas filas. Por exemplo, suponha que o sistema receba os dados brutos das imagens e precise construir as séries temporais calculando o índice de vegetação NDVI de cada *pixel* ao longo do tempo. Antes de enviar os dados para a fila de entrada de processamento dos algoritmos de análise, o stream de dados pode ser enviado para a fila de entrada para o cálculo do NDVI, para depois ser enviado para a fila de processamento dos algoritmos.

Quando os dados chegam a fila de saída do sistema, cabe a um *job* recuperar estes dados e armazenar o resultado no DFS. As aplicações clientes devem ficar constantemente consultando o sistema para recuperar o resultado do processamento dos seus algoritmos. Esses resultados são obtidos do DFS pela API e retornados ao cliente do sistema.

3. Avaliação

Para validar o DStream foi realizada uma avaliação com o algoritmo BFAST, muito utilizado na análise de séries temporais de sensoriamento remoto, e três variações do mesmo em relação ao tempo de resposta e eficácia na detecção de *breakpoints* em uma série temporal de imagens de sensoriamento remoto. O algoritmo BFAST tem como objetivo estimar os pontos de *breakpoints* de uma série temporal, ou seja, os pontos em que a tendência da série temporal muda dentro de um determinado período.

O sistema DStream foi instalado em uma máquina com processador Intel Core i7-2670QM (2.2GHz, 6Mb Cache, 4 núcleos, 8 Threads) e 6GB de memória RAM. Foram realizados testes com 72.361 séries temporais de *pixels*, em que cada série temporal possui 394 valores de NDVI ao longo do tempo de um *pixel*. Para cada uma das 72.361 séries temporais utilizadas no experimento, um especialista da área de sensoriamento remoto do Laboratório de Processamento de Imagens e Geoprocessamento (LAPIG)[1] detectou manualmente, através da visualização de imagens de satélite em diferentes períodos, os pontos ao longo do tempo onde houveram mudanças no tipo de uso de solo para cada *pixel*. Essas séries temporais foram enviadas para a plataforma, simulando um *stream* de dados históricos de imagens de sensoriamento remoto.

Como pode ser visto na Figura 2, o BFAST apresentou tempo de resposta em média de aproximadamente 4 segundos para cada série temporal. Foram implementadas

[1] https://www.lapig.iesa.ufg.br/

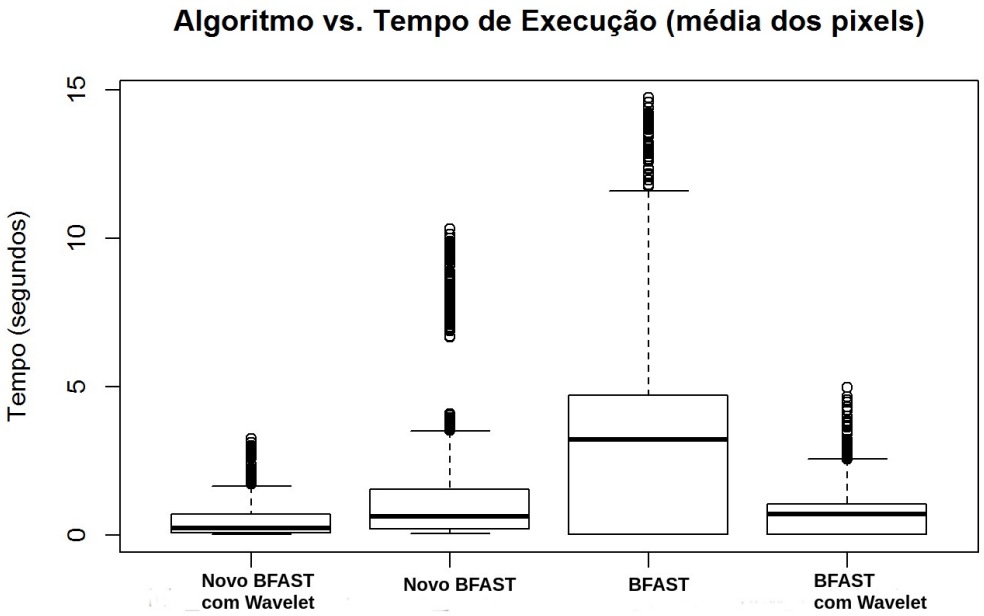

Figura 2. Gráfico de comparação entre as propostas de variação do BFAST.

três variações do BFAST para reduzir o tempo de execução. A primeira variação utiliza Wavelets ("BFAST com Wavelet" na legenda do gráfico da Figura 2), que foi utilizada para reduzir o tamanho da série temporal de 394 valores pela metade, ou seja, para 197 valores, eliminando grande quantidade de ruídos da série temporal. O algoritmo BFAST foi então aplicado sobre essa série temporal de 197 valores e o tempo de execução caiu de 4 segundos para aproximadamente 1,5 segundos. Em relação a eficácia, a utilização de Wavelet reduziu o número de falsos positivos do BFAST, ou seja, reduziu os pontos incorretos no tempo onde o BFAST determina que ocorreram mudanças no tipo de uso de solo através da identificação de *breakpoints*. Isto aconteceu porque com a utilização de Wavelets, o número de ruídos foi reduzido e, assim, mudanças bruscas nos valores do NDVI foram desprezados. Essas mudanças bruscas podem acontecer devido a diversos fenômenos, como a presença de nuvens, que impede que o satélite colete com acurácia suficiente o valor de NDVI do *pixel*. O BFAST foi melhor na identificação dos pontos corretos de mudanças no tipo de uso do solo, pois o uso de Wavelets removeu da série temporal além dos ruídos alguns valores importantes para identificação dos *breakpoints*.

A segunda variação ("Novo BFAST" na legenda do gráfico da Figura 2) modifica o algoritmo do BFAST para utilizar como variável explicativa da regressão linear um valor constante igual a 1, enquanto no algoritmo BFAST a variável explicativa é o tempo. Desta forma, o tempo não tem tanta influência na identificação de *breakpoints*. Esta variação teve desempenho melhor que o BFAST, mas pior que o algoritmo "BFAST com Wavelet". Esta segunda variação conseguiu identificar um maior número de *breakpoints* que as outras técnicas e, por isso, teve uma taxa maior de acertos e de erros.

Uma terceira variação ("Novo BFAST com Wavelet" na legenda do gráfico da Figura 2) executa a modificação do BFAST proposta na segunda variação sobre a série temporal depois de aplicada a Transformada de Wavelet. Ela teve o menor tempo de resposta entre todas as técnicas avaliadas, aproximadamente 1 segundo, tendo uma acurácia média na identificação de falsos positivos e verdadeiros *breakpoints*. Em relação aos fal-

sos positivos teve uma acurácia pior apenas que o "BFAST com Wavelet". Para encontrar os verdadeiros pontos de *breakpoints* teve uma acurácia pior que o BFAST e a segunda variação do BFAST, denominado "Novo BFAST".

4. Conclusão

O grande volume de dados não estruturados, com alta dimensionalidade, sendo gerados em grande quantidade a todo momento coloca o problema de análise de séries temporais no contexto de *Big Data* [Fan et al. 2014]. Para que seja possível explorar o potencial do *Big Data*, este trabalho propõe a plataforma DStream para processamento distribuído de stream de dados de séries temporais, com a execução do algoritmo de análise de séries temporais utilizando várias variações do algoritmo BFAST para validar o casamento da arquitetura da plataforma com o problema de análise de séries temporais.

A plataforma DStream foi capaz de processar as análises de séries temporais com as variações do algoritmo BFAST, processando as séries temporais como *stream* de dados. Assim, a arquitetura pode ser validada em relação ao problema de executar *streams* de dados de sensoriamento remoto. Em trabalhos futuros, serão realizadas avaliações com maior número de máquinas, mas a arquitetura proposta para a plataforma DStream permite que mais máquinas sejam adicionadas tendo uma redução no tempo de resposta final da análise da série temporal.

Referências

Chen, M., Mao, S., and Liu, Y. (2014). Big data: A survey. *Mobile Networks and Applications*, 19(2):171–209.

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.

de Oliveira, S. S. T., de Castro Cardoso, M., dos Santos, W., Costa, P., do Sacramento Rodrigues, V. J., and Martins, W. S. (2017). Distsensing: A new platform for time series processing in a distributed computing environment. *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.

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.

Rakthanmanon, T., Campana, B., Mueen, A., Batista, G., Westover, B., Zhu, Q., Zakaria, J., and Keogh, E. (2013). Addressing big data time series: Mining trillions of time series subsequences under dynamic time warping. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 7(3):10.

Sottile, M. J., Mattson, T. G., and Rasmussen, C. E. (2009). *Introduction to concurrency in programming languages*. CRC Press.