Metodologia para Clusterização em Tempo Real de Fluxos de Dados Climáticos Distribuídos no Contexto de *Big Data*

João Gabriel Lima [1], Nandamudi Vijaykumar [2], Renato Francês[3], Jeremy Gault [4] Robert Devoy [4], Ádamo Santana[1]

[1]Laboratório de Inteligência Computacional e Pesquisa Operacional - Universidade Federal do Pará – UFPA - Belém - Pará - Brasil

[2]Instituto Nacional de Pesquisas Espaciais – INPE – São José dos Campos – SP, Brasil

[3]Laboratório de Planejamento de redes de Alto Desempenho - Universidade Federal do Pará – UFPA - Belém - Pará – Brasil

[4]Coastal and Marine Resources Centre, Environmental Research Institute, University College Cork (UCC), Naval Base, Haulbowline, Cobh, Ireland.

joao.lima@itec.ufpa.br, vijay@lac.inpe.br, rfrances@ufpa.br, j.gault@ucc.ie, r.devoy@ucc.ie, adamo@ufpa.br

Resumo. *Cresce cada vez mais a quantidade de cenários e aplicações que necessitam de processamento e respostas em tempo real e que se utilizam de modelos matemáticos e de mineração de dados a fim de garantir o melhor suporte à tomada de decisão. Neste trabalho, propomos uma metodologia para processamento e clusterização de grandes volumes de fluxos, contínuos e infinitos, de dados com respostas em tempo real, através do uso de redes neurais artificiais conhecidas como mapas auto-organizáveis. Os experimentos e simulações foram realizadas em um ambiente de Cloud Computing sobre um conjunto de dados climáticos. Os resultados mostram a eficiência da proposta ao garantir que o modelo neural utilizado possa gerar respostas em tempo real para o processamento de Big Data.*

1. Introdução

É cada vez mais comum a existência de cenários que requerem o processamento de fluxos de dados em tempo real, onde os dados chegam e uma resposta é executada para reagir à essa nova informação contribuindo imediatamente pra a tomada de decisão. Este tipo de computação é geralmente chamada de *Stream Computing* ou processamento de fluxo de dados [Henrique e Andrade 2012]. No modelo de data mining tradicional, os dados são coletados e o processo de extração do conhecimento é iniciado. Na prática faz-se a garimpagem em cima de dados estáticos que não refletem o momento atual, mas sim o contexto passado. Com *Stream Computing* este processo de mineração de dados pode ser efetuado em tempo real. Em vez de disparar consultas sobre uma base de dados estática, coloca-se uma corrente contínua de dados (*streaming*) atravessando um conjunto de consultas e operações.

Neste trabalho é proposta uma metodologia para a execução da *clusterização* usando o modelo neural de mapas auto-organizáveis sobre fluxo contínuo de grande volume de dados retornando respostas em tempo real.

2. Referencial Teórico

Nesta seção serão apresentados os conceitos básicos das técnicas e ferramentas utilizadas neste trabalho.

2.1 Redis – Banco de dados Não-Relacional

O REmote DIctionary Service – Redis – é um banco de dados de código aberto do tipo chave/valor semi-persistente. Seu melhor uso é em sistemas que exigem rápida troca de dados e muita frequência de escrita, como por exemplo, em aplicações com dados em tempo real, preço das ações, etc [Redmond 2014]. No contexto de *big data*, a utilização desta solução se torna ainda mais eficiente haja vista sua alta capacidade para escalabilidade horizontal.

2.2 Mapas Auto-organizáveis

Os mapas auto-organizáveis fazem parte de um grupo de redes neurais chamado de: redes baseadas em modelos de competição ou, simplesmente, redes competitivas. Outra característica importante deste tipo de rede é que elas utilizam treinamento não-supervisionado, onde a rede busca encontrar similaridades baseando-se apenas nos padrões de entrada. O principal objetivo dos mapas auto-organizáveis - SOM (*self-organizing map)* – é agrupar os dados de entrada que são semelhantes entre si formando classes ou agrupamentos denominados clusters.

2.4 Apache Storm - Processamento de Fluxos de Dados em tempo-real

O Apache Storm é um sistema de processamento de fluxo de dados distribuído e em tempo real com tolerância a falhas, o que garante o processamento eficiente de um fluxo contínuo e ilimitado de dados. Neste *framework*, a abstração de um fluxo de dados é chamada de *Stream*, que é uma sequência ilimitada de tuplas. Uma tupla é uma estrutura que pode representar tipos padronizados de dados (como inteiros, pontos flutuantes, array de bytes) ou tipos definidos pelo usuário através do uso de algum código de serialização. Os *streams* originam-se de estruturas chamadas de *Spouts*. A estrutura responsável por consumir e produzir os dados do fluxo é chamado de *Bolt*.

5. Clusterização de fluxos de dados distribuídos em tempo real

Nesta seção serão descritos os detalhes da solução proposta, sua visão conceitual, arquitetural e implementação.

5.1 Metodologia para a Clusterização em Tempo Real de Fluxos de Dados Climáticos Distribuídos no Contexto de *Big Data*

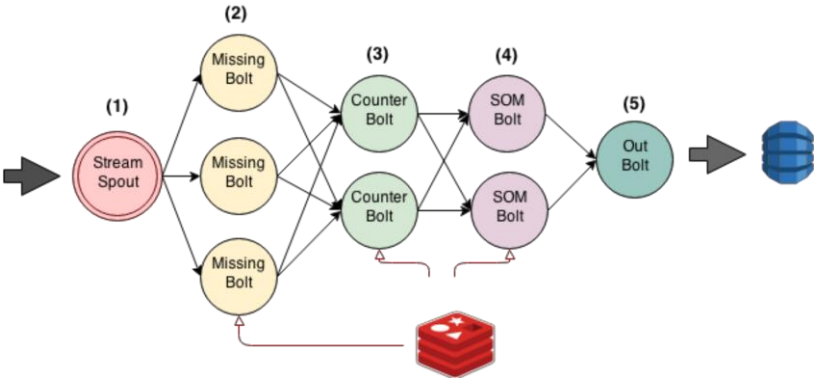
A metodologia utilizada neste trabalho foi desenvolvida para ser uma solução genérica, podendo ser aplicada sobre qualquer domínio de dados e sua estrutura foi baseada para a execução em ambientes distribuídos de *cloud computing*.

Nos itens subsequentes desta seção, serão apresentados os detalhes da arquitetura e funcionamento do processo.

5.1.1 Modelagem da Solução

O Apache Storm, possui uma estrutura chamada de *Spout*, que representa a fonte do fluxo de dados, o primeiro passo para a modelagem da solução é definir uma fonte de dados contínua. Para tal, utilizamos o banco de dados Redis e sobre ele usamos uma abstração de estrutura de dados de filas FIFO (*first-input/first-output*), criando assim, uma fila de dados que são produzidos por uma instância (responsável apenas por produzir o fluxo de dados) e consumidos por outra instância (ou conjunto distribuído de instâncias) em tempo. Na Figura 2, apresentamos uma visão geral, onde em (1) temos a fonte do fluxo de dados em (2) instâncias do MissingBolt recebem as tuplas e verificam sua consistência, em (3) o CounterBolt, recebe as tuplas já validadas e executa o *Map-Reduce*, em (4) o SOMBolt *clusteriza* os dados e em (5) o Outbolt, cuida do armazenamento de resultados e os dispõe para análise e suporte à decisão.

Figura 2. Topologia do StormSOM



Esta topologia conta com 4 nós de processamento, sendo eles (foram dadas nomenclaturas para cada *bolt*, a fim de facilitar a especificação da arquitetura):

1. **MissingBolt**: nó de processamento responsável por verificar a existência de dados faltosos. O **MissingBolt**, trata as tuplas defeituosas (ausência de dados), evitando que ela seja enviada para *clusterização*.
2. **CounterBolt**: nó de processamento responsável por realizar a operação de contagem por frequência, utilizando *map-reduce* [Redmond 2014].
3. **SOMBolt**: nó de processamento responsável pela *clusterização* dos dados propriamente dita. Obtém a tupla já validada pelo **MissingBolt** e registrada para contagem no **CounterBolt** e inicia o fluxo de *clusterização* em 3 etapas:
4. **OutBolt**: por fim, o último nó da topologia é executado. Este *bolt* recebe as tuplas já com a *clusterização*, representada por um campo a mais que corresponde ao cluster a qual a tupla pertence.

Na próxima seção, serão apresentados os ambientes de simulação e os resultados dos testes realizados no StormSOM com a base de dados climáticos, base de estudo deste trabalho.

6. Simulações e Resultados

6.1 Ambiente de simulação

As análises foram feitas utilizando servidores na AWS - Amazon Web Services, com o serviço de Elastic Cloud Compute (Amazon EC2), que permite ao usuário criar instâncias de máquinas com uma variedade de sistemas operacionais de acordo com cada necessidade. Foram utilizadas instâncias que oferecem equilíbrio de recursos de computação, memória e rede.

6.2 Análises de Resultados

Nesta seção serão apresentados os resultados provenientes das simulações realizadas no ambiente de *cloud computing*, onde pretendemos mostrar a eficiência esta metodologia. O objetivo principal desta simulação é a análise quantitativa, haja vista que pretendemos mostrar o desempenho da solução. Em todos os experimentos, foi utilizado um conjunto de 27.280*10^9 registros, enviados em fluxo contínuo e concorrente.

6.2.1 Processamento e Avaliação Quantitativa

Entende-se como um bom resultado, um processamento com baixa-latência, ou seja, o tempo entre a entrada dos dados na topologia de processamento e sua respectiva saída, deve ser mínimo e se manter estável, mesmo diante de variações no fluxo de dados. Este experimento foi realizado utilizando uma instância como nó de processamento e nele pretendemos mostrar o comportamento do algoritmo em relação ao processamento do fluxo de dados(Figura 3).

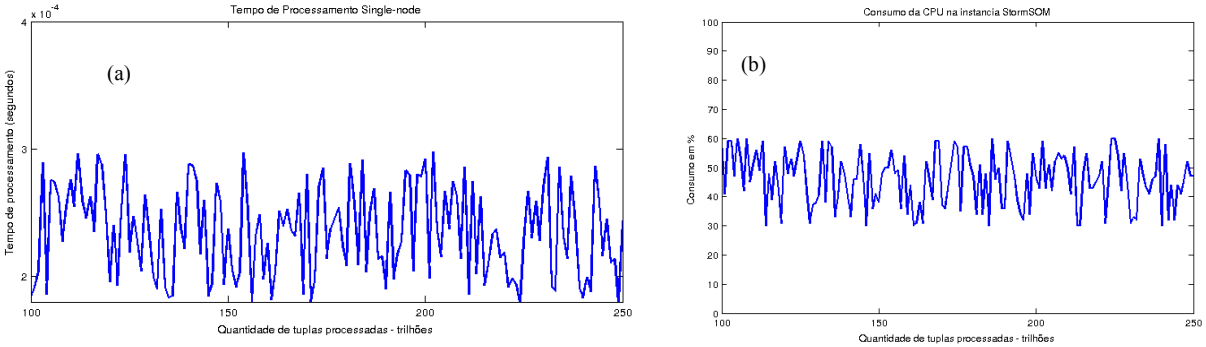


Figura 3. (a) Tempo de Processamento em um único nó. (b) Consumo de recursos da instância em modo *single-node*

Avaliamos a relação entre a quantidade de dados envidados no fluxo e o tempo de processamento. Os resultados mostram que o processamento se mantém estável, em relação ao volume de dados a serem processados, como apresentado na Figura 3-b. Além disso, é importante verificar a demanda de processamento na instância, para analisar o consumo dos recursos da máquina em relação ao uso de CPU. O tempo de processamento total varia entre 0.0000148 e 0.000176 segundos e é possível perceber variações homogêneas durante processamento das tuplas, ocorrendo alguns padrões de picos de processamento. Observou-se que esta variação é ocasionada devido à etapa de leitura e escrita do Redis, pois como seus dados são guardados em memória, e a máquina possui memória limitada. Persistir em blocos, consome menos recursos do banco de dados se compararmos com a persistência de dados unitários, entretanto, pode

prejudicar o tempo de processamento, portanto, é importante atentar para o momento correto que a persistência será unitária ou em bloco.

7. Conclusão

Neste trabalho foi apresentada uma metodologia para *clusterização* distribuída de grandes volumes de fluxos de dados. Como trabalhos futuros, pretendemos expandir as análises de modo a trazer um foco maior para a qualidade dos resultados comparando os mesmos com resultados obtidos através de implementações tradicionais do algoritmo SOM. Por fim, pretendemos refinar a arquitetura proposta com as inovações tecnológicas que surgiram ao longo desta pesquisa.

Agradecimentos

Agradecemos o Instituto de Pesquisas Espaciais - INPE, pela concessão dos dados utilizados para as análises experimentais deste trabalho e pela motivação para a construção de uma solução eficiente para o problema do processamento desses tipos de dados.

Referências

Tom White. Hadoop: The definitive guide. O'Reilly, 2012. 7

Justin Erickson Marcel Kornacker. Cloudera impala: Real-time queries in apache hadoop, for real. http://blog.cloudera.com/blog/2012/10/ cloudera-impala-real-time-queries-in-apache-hadoop-for-real/

Henrique C. M. Andrade, J. P. Morgan Buğra Gedik. Fundamentals of Stream Processing: Application Design, Systems, and Analytics, 2013.

Seguin K. The Little Redis Book. Disponível em https://github.com/karlseguin/the-little-redis-book. Acesso em agosto de 2014.

Redmond, E.; Wilson, J. R. Seven Databases in seven weeks - A Guide to Modern Databases and the NoSQL Movement. Ed. The Pragmatic Programmers, 2014.

Pakhira, M. K., Bandyopadhyay, S., Maulik, U. Validity index for crisp and fuzzy clusters, Pattern Recognition, June 2004.

Simon H. Redes neurais: princípios e técnicas, Porto Alegre: Bookman; 2001.

Joey R. Object-oriented neural networks in C++, London: Academic Press; 1997.

Nathan Marz. Trident tutorial - nathanmarz/storm wiki. Disponível em https://github.com/nathanmarz/storm/wiki/Trident-tutorial

Cascading — application platform for enterprise big data. Disponível em: http://www.cascading.org

Bifet, G. Holmes, R. Kirkby, and B. Pfahringer, "MOA: Massive Online Analysis," J. Mach. Learn. Res., vol. 11, pp. 1601–1604, Aug. 2012.

M. Wojnarski, "Debellor: A Data Mining Platform with Stream Architecture," in Transactions on Rough Sets IX, pp. 405–427, Springer, 2008.