Processamento da Junção Espacial Distribuída utilizando a técnica de Semi-Junção Espacial

Sávio S. Teles de Oliveira[2], Anderson R. Cunha[2], Vagner J. do Sacramento Rodrigues[2], Wellington S. Martins[1]

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

[2]GoGeo
Rua Leopoldo Bulhões, esquina com a Rua 1014
Quadra 31, Lote 07, Sala 9 Setor Pedro Ludovico
CEP 74820-270 – Goiânia – GO – Brazil

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

Abstract. *This paper presents a new Distributed Spatial Join algorithm with Semijoin technique in a scalable peer-to-peer platform. This technique reduces the network traffic and minimize the processing cost. Tests have demonstrated that the response time has been drastically reduced with the Distributed Spatial Join proposed in this paper.*

Resumo. *Este trabalho apresenta um novo algoritmo de Junção Espacial Distribuída utilizando a técnica de Semi-Junção em uma plataforma peer-to-peer escalável. Esta técnica reduz o tráfego de dados na rede e minimiza o custo de processamento. Testes demonstraram que o tempo de resposta foi drasticamente reduzido com a utilização do algoritmo de Junção Espacial Distribuída proposta neste trabalho.*

1. Introdução

A junção espacial é uma das operações mais importantes nos Sistemas de Gerenciamento de Bancos de Dados Espaciais [Zhou et al. 1997] e envolve o relacionamento entre duas bases de dados. Por exemplo: encontrar as rodovias que intersectam com rios (pontes).

Para processar de forma eficiente a operação de junção espacial, as pesquisas têm se concentrado em resolver o problema de forma distribuída utilizando *clusters* de computadores. Com isso, algumas questões no processamento de consultas são identificadas, tais como o tráfego de dados na rede, que impacta de forma significativa no desempenho da Junção Espacial Distribuída.

Este trabalho têm como objetivo apresentar um algoritmo de Junção Espacial Distribuída utilizando a técnica de Semi-Junção Espacial implementado em uma plataforma *peer-to-peer* escalável. Os testes demonstraram que a utilização do algoritmo reduziu drásticamente o tráfego de dados na rede e o tempo de resposta.

O trabalho está organizado da seguinte forma. A Seção 2 descreve as propostas encontradas na literatura para o processamento da junção espacial distribuída. A Seção

3 apresenta o algoritmo de junção espacial distribuído implementado neste trabalho. A Seção 4 descreve a técnica de Semi-Junção proposta neste trabalho. A Seção 5 apresenta os experimentos realizados. A Seção 6 apresenta as conclusões e os trabalhos futuros.

2. Trabalhos Correlatos

Esta Seção apresenta um grupo de trabalhos que processam a junção espacial distribuída utilizando a técnica de Semi-Junção Espacial. Alguns trabalhos encontrados na literatura, como [Mutenda and Kitsuregawa 1999, Patel and DeWitt 2000, Chung et al. 2005, Wei et al. 2008, Zhou et al. 2011, Zhong et al. 2012], processam a junção espacial em um *cluster*, mas não discutem ideias de redução de tráfego de dados na rede utilizando a técnica de Semi-Junção Espacial. Em [Tan et al. 2000] são indexadas as duas bases de dados R e S envolvidas na junção espacial e armazenadas nos servidores R_{site} e S_{site} respectivamente. O trabalho apresentado em [Ramirez and de Souza 2001] utiliza aproximações mais acuradas para processar a junção espacial. Em [Kang and Choy 2002], são propostos alguns modelos de custo para o processamento da junção espacial distribuída. Em [Karam and Petry 2005] são propostos vários modelos de custo detalhados para o processamento da junção espacial distribuída. Estes trabalhos não apresentam estratégias para processar a Junção Espacial Distribuída em mais de dois servidores como em um *cluster* de computadores.

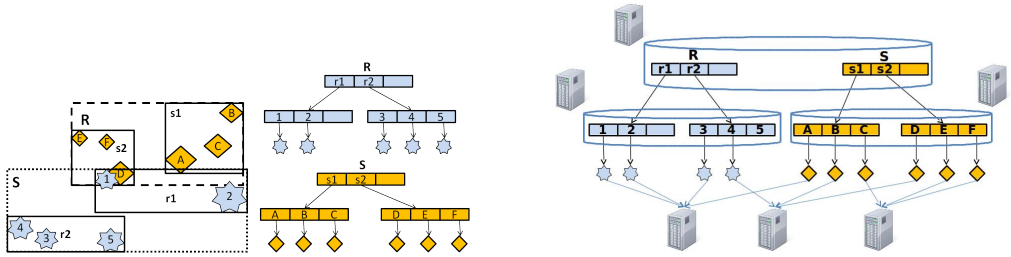
Uma plataforma de processamento da junção espacial distribuída é proposta em [Oliveira et al. 2013], com a necessidade de um serviço de nomes para descobrir qual máquina armazena cada nó da *R-Tree* distribuída. Desta forma, a junção espacial é muito penalizada pela necessidade de consulta ao serviço de nomes a cada passo do algoritmo. Além disso, o algoritmo de Semi-Junção transfere os objetos de apenas uma relação, ao invés de analisar cada tupla da Junção para decidir qual objeto transferir. Em [Farruque and Osborn 2014] é proposto um algoritmo de Semi-Junção com vários servidores particionando os índices espaciais ou através de uma representação com Bloom Filter. O algoritmo trafega na rede a base de dados da junção com menor cardinalidade. Entretanto, a base de dados de menor cardinalidade pode conter polígonos com geometrias com grande quantidade de pontos geográficos, o que aumentaria o tráfego de dados.

3. Processamento da junção espacial distribuída

A junção espacial pode ser definida a partir de duas relações R = $r_1, ..., r_n$ e S = $s_1, ..., s_m$, onde r_i e s_j são objetos espaciais, $1 \leq i \leq n$ e $1 \leq j \leq m$. A operação verifica todos os pares (r_i, s_j) que satisfazem o predicado de um operador topológico, por exemplo a interseção, isto é, $r_i \bigcap s_j \neq \emptyset$.

O processamento da junção é realizado em duas etapas: etapa de filtragem e etapa de refinamento [Patel and DeWitt 1996]. A etapa de filtragem inicia na raiz das duas relações R e S e é realizada nos nós internos da *R*-Tree*. Esta etapa gera um conjunto de possíveis respostas a consulta. A fase de refinamento é realizada nas folhas e remove deste conjunto os resultados incorretos utilizando as geometrias reais de cada objeto.

No exemplo da Figura 1(a), a etapa de filtragem analisa as raízes de R e S e a etapa de refinamento analisa os nós filhos de (r1, s1) e (r1, s2), pois estes apresentaram intersecção entre os seus respectivos MBRs na fase de filtragem. Apenas (1, D) fez parte do resultado por apresentar intersecção de suas respectivas geometrias.



(a) Visão Espacial das relações R e S (b) R*-Tree Distribuída

Figura 1. Junção Espacial

Na versão distribuída do algoritmo de junção espacial, há a necessidade de trocar mensagens na rede para acessar os objetos distribuídos. O algoritmo de junção espacial distribuída deste trabalho foi implementado utilizando a plataforma DistJoin. Esta plataforma possui uma arquitetura *peer-to-peer* escalável descentralizada com tolerância a falhas e alta disponibilidade. O protocolo *Gossip* é utilizado para disseminar informações do *cluster* e de localização dos nós da *R-Tree* para que não seja necessário um Serviço de Nomes para descobrir a localização de um nó distribuído durante a execução do algoritmo de Junção Espacial Distribuída.

A Figura 1(b) ilustra as *R*-Trees* de R e S distribuídas em um *cluster* de computadores. Quando dois objetos espaciais estão localizados em máquinas diferentes, por exemplo os objetos 1 e D, um deles deve ser trafegado na rede até o local em que está armazenado o outro objeto. O tráfego de dados na rede é reduzido neste trabalho utilizando uma técnica conhecida como Semi-Junção espacial, apresentada na Seção 4.

4. Algoritmo de Semi-Junção Espacial

Este trabalho apresenta o algoritmo de Semi-Junção Espacial executado entre cada par de objetos (r, s) das bases R e S na etapa de refinamento da Junção Espacial Distribuída. Para reduzir o tráfego de dados na rede, o objeto espacial com maior número de pontos será visto como b e o outro como a, já que a aproximação do polígono de b e o polígono de a serão transferidos pela rede. O número de pontos do polígono de cada objeto espacial e é armazenado como metadado no nó pai de e. Na Semi-Junção Espacial, o par de tuplas a e b estão localizados nos servidores A_{server} e B_{server} respectivamente.

O algoritmo segue três passos. O passo 1 do algoritmo envia a aproximação do polígono de b, denominada b', para A_{server}. Neste trabalho, o MBR é utilizado como aproximação da geometria. No passo 2, é realizada a junção entre b' e o polígono a em A_{server}. A aproximação têm menos pontos que a geometria real, o que faz com que a computação geométrica da junção entre b' e a tenha processamento minimizado, já que o custo dos algoritmos espaciais são proporcionais ao número de pontos das geometrias.

As aproximações filtram apenas os resultados que não fazem parte da resposta. Por isso, é retornado o polígono de a para B_{server}, caso b' e a apresentem intersecção e vazio, caso contrário. O algoritmo de intersecção é executado, no passo 3, entre os polígonos de a e b para verificar se apresentam intersecção. Caso apresentem, os dois objetos são retornados como um dos pares de resultados da consulta.

No exemplo da Figura 2 com o par de objetos (1, D), o polígono 1 contém mais

pontos que D e, por isso, é enviada sua aproximação, no passo 1, e retornado o polígono D como resposta no passo 2. Assim, o objeto 1 é definido como b e D como a.

O algoritmo de Semi-Junção Espacial consegue reduzir o tráfego de dados na rede, através da transmissão do objeto espacial que apresenta geometria com menos pontos. Além disso, é minimizado o processamento local, pois no passo 2 do algoritmo é realizada a intersecção entre a aproximação da geometria com maior número de pontos (b') e a geometria com menor número de pontos (a). Se estas geometrias não apresentarem intersecção, não será preciso realizar a intersecção entre a e b.

5. Experimentos

Foram utilizadas as seguintes bases de dados[1]: i) 10994 polígonos do Bioma da Caatinga com tamanho total de 275 MB, ii) 21840 pontos de Localidades do Brasil com tamanho total de 1,4 MB, iii) 5771 linhas de Hidrografia do Brasil com tamanho total de 1,4 MB e iv) 5565 polígonos de municípios do Brasil com tamanho total de 38 MB. Foram executados as seguintes junções espaciais: i) Bioma da Caatinga e Localidades que retorna 3934 itens e ii) Hidrovia e Municípios que retorna 8721 itens.

A execução dos testes de junção espacial distribuída têm como objetivo avaliar os seguintes aspectos: i) a escalabilidade do algoritmo de junção espacial distribuída; ii) o tráfego de dados na rede. Os testes foram executados com e sem a técnica de Semi-Junção Espacial em um *cluster* com 3, 6 e 12 servidores, com cada máquina contendo um processador Optiplex 780 Intel Core 2 Quad 2.83GHz com 4 Gb de memória RAM conectadas por uma rede Ethernet de 1 Gbit/segundo.

Como pode ser visto na Figura 2(a), a junção espacial entre as bases de dados de Bioma da Caatinga e Localidades, apresentou um tempo de execução drasticamente menor utilizando a técnica de Semi-Junção Espacial proposta neste trabalho, sendo 5 vezes melhor na média que os testes executados sem a técnica.

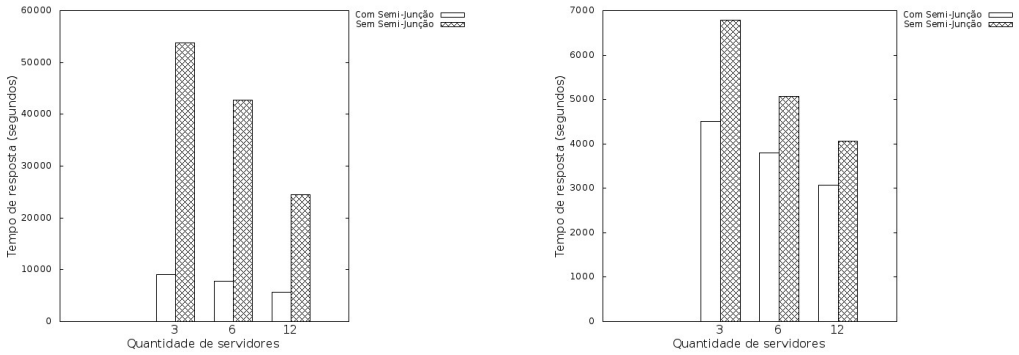
Isto ocorreu devido a redução do tráfego de dados na rede que foi aproximadamente 9 vezes menor com 3 máquinas, 11 vezes menor com 6 servidores e 19 vezes menor com 12 servidores utilizando a técnica de Semi-Junção Espacial.

A junção espacial entre as bases de dados de Municípios e Hidrovia apresentou tempo de resposta aproximadamente 1,5 vezes melhor com a técnica de Semi-Junção Espacial deste trabalho, como pode ser visto na Figura 2(b). O tráfego de dados na rede nesta junção foi reduzido em aproximadamente 1,3 com a técnica de Semi-Junção Espacial para 3, 6 e 9 servidores.

O tempo de resposta nesta junção não foi reduzido na mesma proporção que a junção apresentada na Figura 2(a), pois o tráfego de dados na rede sem a técnica de Semi-Junção é menor na junção da Figura 2(b) devido ao menor espaço em disco de Municípios em relação a Bioma da Caatinga.

O algoritmo de Junção Espacial Distribuída proposto neste trabalho foi aproximadamente quatro vezes melhor que a proposta de [Oliveira et al. 2013], já que a plataforma de [Oliveira et al. 2013] necessita consultar o Serviço de Nomes a cada acesso a um nó da

[1]Bases de dados disponibilizadas pelo Laboratório de Processamento de Imagens e Geoprocessamento - www.lapig.iesa.ufg.br



(a) Bioma da Caatinga e Localidades.

(b) Municípios e Hidrovia.

Figura 2. Tempo de resposta da Junção Espacial Distribuída.

R-Tree Distribuída. Além disso, a Semi-Junção proposta em [Oliveira et al. 2013] sempre envia os objetos da relação com menor número de pontos pela rede. Nosso trabalho analisa cada par de objetos individualmente.

O algoritmo de Junção Espacial Distribuída apresentou escalabilidade em todos os testes executados. A utilização da técnica de Semi-Junção Espacial proposta neste trabalho reduziu o tráfego de dados na rede e o tempo de resposta da Junção Espacial Distribuída.

6. Conclusões e Trabalhos Futuros

A junção espacial apresenta alto custo e pode ser processada de forma eficiente em um *cluster* de computadores para distribuir o processamento da operação. Para processar a Junção Espacial Distribuída de forma eficiente, o tráfego de dados na rede deve ser reduzido. Para tal, existe uma técnica denominada Semi-Junção Espacial. Nenhum trabalho encontrado na literatura, entretanto, processa a Junção Espacial Distribuída utilizando uma técnica de Semi-Junção Espacial eficiente em uma plataforma *peer-to-peer* escalável.

Por isso, neste trabalho foi implementado um algoritmo de processamento da Junção Espacial Distribuída utilizando uma técnica de Semi-Junção Espacial para ser processado em um *cluster* de computadores utilizando uma plataforma *peer-to-peer*. O algoritmo se apresentou escalável nos experimentos e a utilização da técnica de Semi-Junção Espacial reduziu de forma significativa o tráfego de dados na rede e o tempo de resposta.

Em trabalhos futuros, além do número de pontos das geometrias, novos metadados serão adicionados (i.e. CPU, memória, rede), para decidir quais geometrias serão transferidas e onde elas serão processadas. Também serão analisadas novas aproximações na técnica de Semi-Junção Espacial. Novos experimentos serão realizados com outras bases de dados para validar o algoritmo de Junção Espacial Distribuída em diversos cenários.

Referências

Chung, W., Park, S., and Bae, H. (2005). Efficient parallel spatial join processing method in a shared-nothing database cluster system. *Embedded Software and Systems*, pages 81–87.

Farruque, N. and Osborn, W. (2014). Efficient distributed spatial semijoins and their application in multiple-site queries. In *Advanced Information Networking and Applications (AINA), 2014 IEEE 28th International Conference on*, pages 1089–1096. IEEE.

Kang, M. and Choy, Y. (2002). Deploying parallel spatial join algorithm for network environment. In *High Speed Networks and Multimedia Communications 5th IEEE International Conference on*, pages 177–181. IEEE.

Karam, O. and Petry, F. (2005). Optimizing distributed spatial joins using r-trees. In *Proceedings of the 43rd annual Southeast regional conference-Volume 1*, pages 222–226. ACM.

Mutenda, L. and Kitsuregawa, M. (1999). Parallel r-tree spatial join for a shared-nothing architecture. In *Database Applications in Non-Traditional Environments, 1999.(DANTE'99) Proceedings. 1999 International Symposium on*, pages 423–430. IEEE.

Oliveira, S., Sacramento, V., Cunha, A., Aleixo, E., de Oliveira, T., Cardoso, M., and Junior, R. (2013). Processamento Distribuído de Operações de Junção Espacial com Bases de Dados Dinâmicas para Análise de Informações Geográficas. *XXXI Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*.

Patel, J. and DeWitt, D. (1996). Partition based spatial-merge join. In *ACM SIGMOD Record*, volume 25, pages 259–270. ACM.

Patel, J. and DeWitt, D. (2000). Clone join and shadow join: two parallel spatial join algorithms. In *Proceedings of the 8th ACM international symposium on Advances in geographic information systems*, pages 54–61. ACM.

Ramirez, M. and de Souza, J. (2001). Distributed processing of spatial join. In *Proc. of the Anais do III Workshop Brasileiro de GeoInformática GeoInfo*, volume 2001, pages 1–8.

Tan, K., Ooi, B., and Abel, D. (2000). Exploiting spatial indexes for semijoin-based join processing in distributed spatial databases. *Knowledge and Data Engineering, IEEE Transactions on*, 12(6):920–937.

Wei, H., Wei, Z., and Yin, Q. (2008). A new parallel spatial query algorithm for distributed spatial databases. In *Machine Learning and Cybernetics, 2008 International Conference on*, volume 3, pages 1570–1574. IEEE.

Zhong, Y., Han, J., Zhang, T., Li, Z., Fang, J., and Chen, G. (2012). Towards parallel spatial query processing for big spatial data. In *Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2012 IEEE 26th International*, pages 2085–2094. IEEE.

Zhou, X., Abel, D., and Truffet, D. (1997). Data partitioning for parallel spatial join processing. In *Advances in Spatial Databases*, pages 178–196. Springer.

Zhou, Y., Zhu, Q., and Zhang, Y. (2011). Spatial data dynamic balancing distribution method based on the minimum spatial proximity for parallel spatial database. *Journal of Software*, 6(7):1337–1344.