Processamento Distribuído da Junção Espacial de Múltiplas Bases de Dados - *Multi-way Spatial Join*

Anderson R. Cunha[1], **Sávio S. T. de Oliveira**[1], **Everton L. Aleixo**[2], **Marcelo de C. Cardoso**[2], **Thiago B. de Oliveira**[1], **Vagner J. do Sacramento Rodrigues**[2]

[1]Instituto de Informática - INF – Universidade Federal de Goiás - UFG
Bloco IMF I - Campus II – Samambaia – Caixa Postal 131 – CEP 74001-970 - Goiânia – GO

[2]Departamento de P&D – goGeo.io
Alameda Leopoldo de Bulhões, Número 30 - Setor Pedro – Goiânia – GO

{andersoncunha,savioteles,thborges}@ufg.br

{everton.lima,marcelo.cardoso,vagner}@gogeo.io

Abstract. *This paper proposes the Distributed Synchronous Traversal algorithm (DST) for distributed processing of spatial join with multiple datasets (Multi-way Spatial Join). It was evalueted over clusters up to eigth servers, and just like the Synchronous traversal algorithm, performs spatial join by synchronously traversing the input R-Trees. The experiments results on real datasets have shown that the algorithm provides a satisfactory degree of horizontal scalability.*

Resumo. *Neste trabalho é proposto o algoritmo Distributed Synchronous Traversal (DST) para o processamento distribuído da junção espacial com múltiplas bases de dados (Multi-way Spatial Join). Ele foi avaliado sobre clusters de até oito servidores, e assim como o algoritmo Synchronous Traversal (ST), realiza o processamento da junção espacial percorrendo de forma síncrona os índices R-Tree das bases de dados de entrada. Os resultados gerados mostraram que o algoritmo apresentou um grau de escalabilidade horizontal satisfatório.*

1. Introdução

A junção espacial (*Spatial Join*) é uma das principais operações espaciais para oferta de serviços em aplicações de *Location Based Services* (LBS), e também é uma das que mais demanda recurso computacional [Mutenda and Kitsuregawa 1999] para cruzamento de duas bases de dados georreferenciadas. Sua complexidade e custo computacional aumentam significativamente quando a junção envolve múltiplas bases de dados (*multi-way spatial join*) sobre bases de dados com grande volume de dados georreferenciados (*spatial big data*).

Através de operações de *multi-way spatial join* é possível responder perguntas complexas e de alto impacto cruzando diversas bases de dados simultaneamente. Por exemplo, quais desmatamentos (base de desmatamento) no Brasil estão a 200 metros de rios (base de hidrografia) a 500 metros de áreas de preservação ambiental (base de preservação), 1000 metros de rodovias (base de rodovias) e estão em biomas do cerrado ou mata atlântica (base de biomas)?

Estratégias tradicionais de processamento do *multi-way spatial join* geralmente aplicam combinações de algoritmos de junção espacial binária (*pairwise join algorithms*)

sobre ambientes computacionais centralizados. Em consultas complexas ou sobre grandes volumes de dados, esse tipo de abordagem exige servidores com grande capacidade computacional, muitas vezes inviáveis técnica e financeiramente em ambientes centralizados.

Neste trabalho é proposto um algoritmo distribuído para processamento de operações de *multi-way spatial join* em um *cluster* de computadores. O algoritmo é chamado *Distributed Synchronous Traversal* (DST), se baseia no algoritmo *Synchronous Traversal* (ST) [Papadias et al. 1998, Papadias et al. 1999, Mamoulis and Papadias 2001] e processa o *multi-way spatial join* percorrendo de forma síncrona os índices espaciais de entrada sem geração de bases de dados intermediárias (i.e., o resultado é construído conforme os níveis são percorridos, sem etapas adicionais de persistência e/ou reindexação).

O DST foi implementado e avaliado através de experimentos sobre *clusters* com até 8 servidores em função de dois fatores importantes na execução distribuída do *multi-way spatial join*: ***i)*** a capacidade de escalabilidade horizontal; ***ii)*** a distribuição de carga do processamento.

Até onde conhecemos, não há outras propostas na literatura de algoritmo baseado em uma plataforma *peer-to-peer* para o processamento distribuído de operações de *multi-way spatial join*. Nesse contexto, podemos resumir as contribuições deste trabalho da seguinte forma:

- Projeto e implementação de um algoritmo distribuído para processamento do *multi-way spatial join* de forma escalável em um *cluster* de computadores;
  - Definição de uma estrutura para mapeamento da semântica da consulta e coordenação do processamento distribuído da operação;
  - Algoritmos de restrição espacial para diminuir o número total de operações de junção espaciais binárias e o tráfego de dados na rede.
- Avaliação e experimentação do DST segundo diferentes critérios de análise com bases de dados reais.

2. Processamento de Dados Espaciais

Diversas pesquisas foram realizadas para desenvolver métodos de acesso e estruturas de indexação que fossem adequadas às características dos dados espaciais (multidimensionais) [Bentley 1975, Guttman 1984, Kamel and Faloutsos 1994]. Entre estas estruturas de indexação de dados espaciais, a *R-tree* [Guttman 1984] e suas variações são as mais citadas na literatura e utilizadas em bancos de dados geográficos, tais como Oracle Spatial, PostGIS e MySQL Spatial.

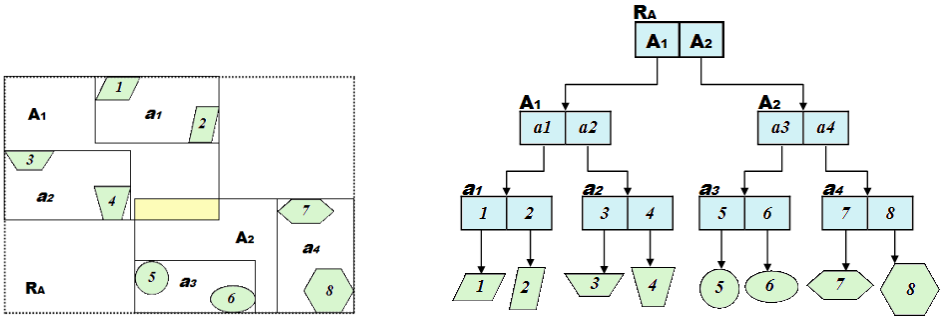
2.1. *Estrutura de Indexação (R-Tree)*

A *R-Tree* é uma estrutura hierárquica que utiliza retângulos para organizar objetos espaciais de forma que objetos colocalizados (i.e., espacialmente próximos) fiquem armazenados próximos uns dos outros, promovendo assim uma redução no espaço de busca a cada nível da árvore. A *R-Tree* é uma árvore balanceada por altura com ponteiros para os objetos espaciais nos nós folhas (ela pode ser vista como uma extensão da B^+*-Tree* [Comer 1979] no espaço multidimensional).

Os retângulos utilizados pelas *R-Trees* são denominados de *Minimum Bounding Rectangles* (MBRs) . Um MBR possui a menor área retangular possível para englobar

completamente seus objetos espaciais. A Figura 1(a) retrata o desenho dos MBRs agrupando os objetos espaciais de 1 a 8 em subconjuntos, de acordo com sua localização espacial.

A Figura 1(b) ilustra a estrutura hierárquica da *R-Tree* apresentando um nó raiz (R_A), nós diretórios (A_1 e A_2) e um último nível com nós folhas (a_1 até a_4). Cada nó armazena no máximo M e no mínimo $m \leq \frac{M}{2}$ entradas. Entre as várias extensões propostas para a *R-Tree*, a R*-Tree propõe mecanismos para reduzir a área morta e as áreas de sobreposição entre os MBRs melhorando o tempo de busca em bases de dados dinâmicas [Beckmann et al. 1990].



(a) Disposição espacial dos objetos geográficos

(b) Índice espacial *R-Tree*

Figura 1. *R-Tree*, representações e estruturas

2.2. *Multi-way Spatial Join*

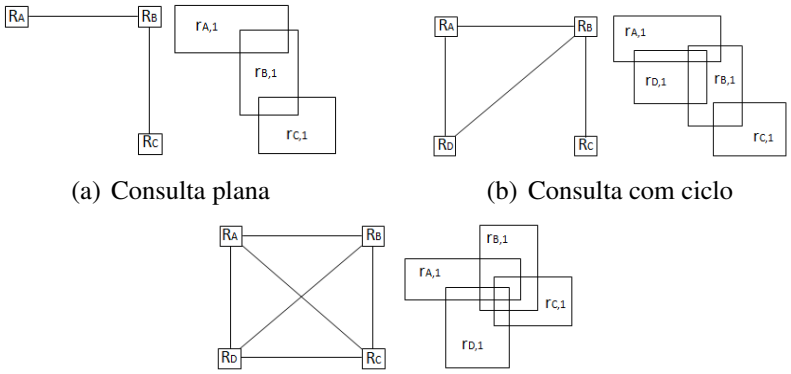
A junção espacial sobre um número arbitrário de bases de dados (*multi-way spatial join*) pode ser formalmente definida da seguinte maneira: dado um conjunto de *n* bases de dados $R_1, R_2, ..., R_n$ e uma consulta Q, onde Q_{ij} é o predicado espacial que deve ser aplicado entre R_i e R_j, retornar todas as *n-uplas* $\{(r_{1,w}, ..., r_{i,x}, ...r_{j,y}, ...r_{n,z}) \,|\, \forall i, j \,:\, r_{i,x} \in R_i,\, r_{j,y} \in R_j \text{ e } r_{i,x}\, Q_{ij}\, r_{j,y}\}$ [Mamoulis and Papadias 2001]. Neste trabalho, a operação de intersecção de geometrias foi utilizada como predicado espacial Q_{ij}.

A consulta Q pode ser vista como um grafo $G = (N, E)$, onde *N* representa o conjunto de bases de dados de entrada, e *E* representa os predicados de junção, assim como em uma rede de restrições (*constraint network*) [Papadias et al. 1999], onde os vértices correspondem as variáveis do problema, e as arestas a restrições espaciais binárias. Essa rede de restrições foi denominada Grafo de Consulta e apresentado em maiores detalhes a seguir.

A Figura 2 apresenta exemplos de consultas *Q*, e possíveis soluções representadas por retângulos. A Figura 2(a) apresenta uma consulta plana com as restrições: $r_{A,1} \cap r_{B,1} \neq \emptyset$ e $r_{B,1} \cap r_{C,1} \neq \emptyset$. A Figura 2(b) apresenta uma consulta com ciclo com as restrições: $r_{A,1} \cap r_{B,1} \neq \emptyset$, $r_{B,1} \cap r_{C,1} \neq \emptyset$, $r_{B,1} \cap r_{D,1} \neq \emptyset$ e $r_{A,1} \cap r_{D,1} \neq \emptyset$. E, por fim, a Figura 2(c) apresenta uma consulta completa (clique), com a restrição: $r_{A,1} \cap r_{B,1} \cap r_{C,1} \cap r_{D,1} \neq \emptyset$.

2.3. Algoritmo *synchronous traversal* (ST)

O algoritmo *Synchronous Traversal* [Papadias et al. 1998, Papadias et al. 1999, Mamoulis and Papadias 2001], que serviu de base para a implementação do DST,



(a) Consulta plana (b) Consulta com ciclo

(c) Consulta clique

Figura 2. Exemplos de Consultas com múltiplas bases de dados

processa o *Multi-way Spatial Join* através da análise de todos os índices de entrada, seguindo combinações de nós que satisfazem as restrições de consulta, de forma similar ao que acontece com o RJ (*R-Tree Join*) [Brinkhoff et al. 1993].

Considere o exemplo de análise de múltiplas bases de dados espaciais, onde um ambientalista pretende identificar possíveis focos de assoreamento em rios que passam por áreas de conservação ambiental. Para tanto, ele tem que fazer a correlação entre as bases de dados espaciais de Desmatamento, Hidrografia e Áreas de conservação ambiental da região de interesse.

Essas bases de dados e os seus respectivos índices *R-Tree* são apresentados na Figura 3. Neste exemplo, a analise *multi-way spatial join* pode ser mapeada em uma consulta plana (como a consulta mostrada na Figura 2(a)) que busca pelo conjunto de todas as 3-uplas (ou triplas) de desmatamento, rios e áreas de conservação que atendem a seguinte restrição: desmatamentos que intersectam com rios que passam por áreas de conservação ambiental.

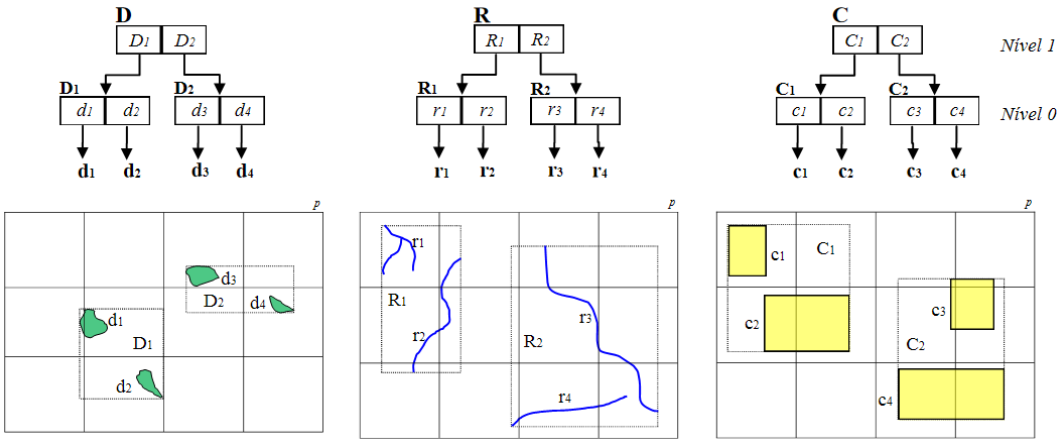


Figura 3. Representação gráfica das bases de dados Desmatamento, Rios e Áreas de Conservação sobre o plano cartesiano, e os respectivos índices R-TRee.

ST realiza o processamento da junção espacial em duas fases, filtragem e refina-

mento: Na fase de filtragem, o ST inicia o processamento nas raízes das *R-Trees*. Onde resolve um único subproblema, ou seja, checagem da combinação dos nós raízes das *R-Tree* a fim de encontrar as candidatas a soluções para processamento no próximo nível. Este processo se repete recursivamente através dos níveis intermediários até que o nível das folhas (nível 0) seja atingido, e se inicie a fase de refinamento com o processamento do resultado final da junção.

No exemplo, das oito combinações possíveis no nível das raízes das *R-Trees* (i.e., $(D_1, R_1, C_1), (D_1, R_1, C_2), ..., (D_2, R_2, C_2)$), apenas $(D_1, R_1, C_1), (D_1, R_2, C_2)$ e (D_2, R_2, C_2) podem levar a solução final e constituem o resultado da fase de filtragem. As demais combinações são descartadas por não atenderem às restrições da consulta.

Na fase de refinamento, o processamento das combinações formadas pelos filhos da candidata a solução (D_1, R_1, C_1) leva à tripla (d_1, r_2, c_2) que faz parte do resultado final. A candidata a solução (D_2, R_2, C_2) leva a outras duas triplas (d_3, r_3, c_3) e (d_3, r_3, c_4) que completam o resultado final da consulta (a tripla (D_1, R_2, C_2) é um falso positivo).

3. Processamento Distribuído da Junção Espacial com Múltiplas Bases de Dados

Para o processamento distribuído do DST foram utilizados servidores em um cluster onde se encontra um índice espacial *R-Tree* Distribuído [de Oliveira et al. 2013], conforme Figura 4, que aliado a mecanismos para o tratamento de concorrência, consistência e um serviço de descoberta possibilitam a paralelização e distribuição do processamento de operações espaciais.

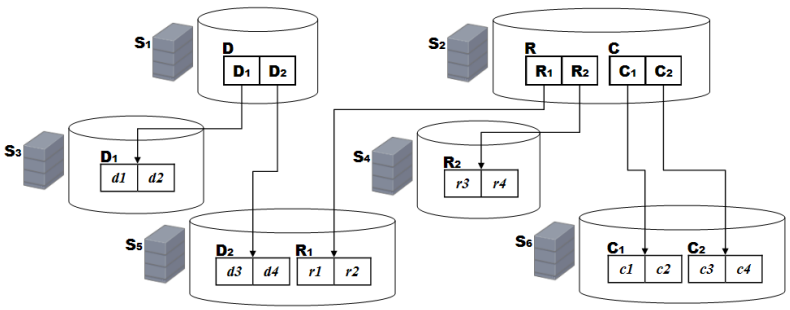


Figura 4. *R-Trees* das bases de dados da Figura 3 distribuídas entre 6 Servidores

As requisições de operações espaciais são encaminhadas para o servidor do *cluster* que contém os dados. O servidor que recebe a requisição necessita, eventualmente, buscar dados em outros servidores ou mesmo repassar para outro servidor a requisição da operação. A troca de dados entre os servidores é minimizada através da escolha da estratégia de distribuição de dados. Neste trabalho, foram experimentados dois algoritmos de distribuição conforme descrito na Seção 4.1.

3.1. *Distributed Synchronous Traversal* (DST)

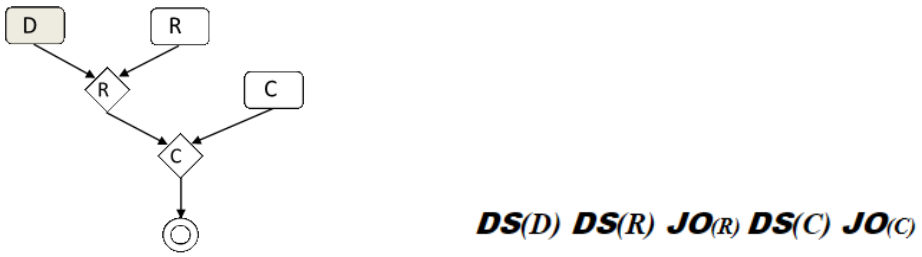
O algoritmo DST adota a abordagem de busca sistemática (Systematic Search Algorithms), como os algoritmos de *backtracking* utilizados em problemas de satisfação de restrições binárias (*binary cosntraint satisfaction problem* - BCSP) [Prosser 1993]. Segundo Bacchus e Van Beek [Bacchus and van Beek 1998], um problema BCSP é definido

sobre um conjunto finito de variáveis n, cada uma com um domínio Δ_i também finito de valores potenciais e uma coleção finita de restrições.

O problema de multi-way spatial join pode ser reduzido para um problema BCSP, uma vez que o conjunto de variáveis n vai ser composto pelos nós das *R-Tree* de entrada, os domínios Δ_i são os filhos dos respectivos nós das *R-Trees* e as restrições binárias C_{ij} entre os pares de variáveis (v_i, v_j) são as arestas do grafo de consulta.

Logo o DST, assim como os algoritmos de *backtracking*, processa a consulta nível a nível. O conjunto de soluções em níveis intermediários das *R-Trees* definem os parâmetros para novos problemas BCSP nos níveis inferiores, que serão processados recursivamente até que no nível das folhas. As soluções de todos os problemas BCSP compõem o resultado final da consulta e são retornados (A Figura 6 ilustra esse comportamento).

Esse processamento realizado pelo DST é semelhante ao processamento realizado pelo ST (vide Seção 2.3), só que de forma distribuída sobre um *cluster* de computadores, conforme ilustrado na Figura 4. Para distribuir o processamento do *multi-way spatial join* pelas máquinas do cluster foi necessário definir um Plano de Consulta que viabilizasse a criação de um Plano de Execução que pode ser executado de forma coordenada e distribuída sobre o *cluster* de computadores.



(a) Plano de Consulta (b) Plano de Execução

Figura 5. Exemplo de Plano de Consulta e Plano de Execução resultante.

A consulta Q deve ser fornecida para o algoritmo na forma de uma árvore binária[1] que por convenção recebe o nome de Plano de Consulta (Figura 5(a)). Esta árvore é percorrida em um percurso de pós-ordem para gerar o Plano de Execução que é uma estrutura baseada em expressões pós-fixas (Figura 5(b)) e será utilizado pelo DST para coordenar o processamento distribuído da consulta.

O Plano de Execução usa a seguinte notação $DS(X)$ para definir as *R-Trees* de entrada (*dataset*) e $JO_{(Y)}$ para representar as operações de junção espacial (operações), tal que X representa o nome da base de dados de entrada (retângulos da Figura 5(a)) e Y compreende os resultados da operação de junção JO (losangos da Figura 5(a)). Esse resultado da operação $JO_{(Y)}$ entra como argumento para a próxima operação de junção, e recebe o nome de $JR_{(Y)}$ (ver Figura 6).

[1] A conversão do Grafo de Consulta para o Plano de Consulta não está no escopo desse trabalho por lidar com algoritmos de otimização dos planos de consulta.

3.2. Processamento do Plano de Execução

No algoritmo DST, todas as operações de junção espacial ($JO_{(X)}$) do Plano de Execução serão executadas no servidor do *cluster* que contém os dados do primeiro argumento da operação. Esta característica aumenta o potencial de escalabilidade do sistema, pois distribui o processamento entre os vários servidores do *cluster*.

Um Plano de Execução $PE_{(k,n)}$ implica no processamento de uma n-upla n no nível k da *R-Tree*. A Figura 6 apresenta, como exemplo, a instância de processamento do Plano de Execução $PE_{(1,0)}$ (Figura 5(b)) sobre as bases de dados da Figura 3. O processamento de $PE_{(1,0)}$ irá resultar em $JR_{(C)}$ contendo as triplas $(D_1, R_1, C_1), (D_2, R_2, C_2)$ e (D_1, R_2, C_2). Essas triplas serão convertidas nos Planos de Execução $PE_{(0,0)}$, $PE_{(0,1)}$ e $PE_{(0,2)}$ que serão redistribuídos pelos servidores do *cluster*.

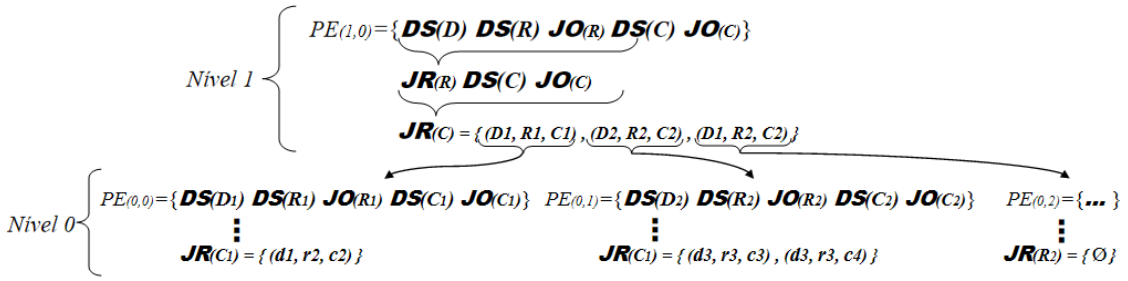


Figura 6. Instância do processamento do plano de execução da Figura 5(b)

No nível das folhas (fase de refinamento), o Plano de Execução $PE_{(0,0)}$ resulta na tripla (d_1, r_2, c_2) e o Plano de Execução $PE_{(0,1)}$ resulta nas triplas (d_3, r_3, c_3) e (d_3, r_3, c_4), estas 3 triplas são retornadas por constituirem o resultado final da consulta. O $PE_{(0,2)}$ é um falso positivo, pois não há intersecção entre as filhos dos nós D_1 e R_2 (ver Figura 3).

4. Experimentos

Os experimentos foram realizados sobre bases de dados reais fornecidas pelo LAPIG[2]. As bases de dados utilizadas e suas características são descritas a seguir: ***1)*** Bioma do Cerrado (*B*), 151986 Polígonos e 411,3MB; ***2)*** Desmatamento do Cerrado (*D*), 32578 Polígonos e 11.2MB; ***3)*** Ferrovias do Brasil (*F*), 617 Linhas e 668KB; ***4)*** Hidrografia do Brasil (*H*), 226963 Linhas e 64,5MB; ***5)*** Hidrovias do Brasil (*I*), 5571 Linhas e 1.4MB; ***6)*** Localidades (*L*), 21840 Pontos e 1,4MB; ***7)*** Municípios do Brasil (*M*), 5564 Polígonos e 38.8MB; ***8)*** Rodovias (*R*), 51645 Linhas e 15,2MB; e ***9)*** Vegetação do Brasil (*V*), 2140 Polígonos e 4.7MB.

Estas bases de dados foram combinadas em consultas planas (ver Figura 2(a)) com 3 e 4 entradas, variando o tamanho das bases de dados, a quantidade de entradas, e o tipo de geometria processada. Abaixo são descritos os planos de consulta, incluindo o Plano de Execução equivalente e o número de resultados gerados por cada uma:

- ***1-*** Hidrovias do Brasil x Hidrografia do Brasil x Rodovias do Brasil x Ferrovias do Brasil (***IHRF***, 285 resultados):
  - $DS(I)\ DS(H)\ JO_{(H)}\ DS(R)\ JO_{(R)}\ DS(F)\ JO_{(F)}$;

[2]Laboratório de Processamento de Imagens e Geoprocessamento (www.lapig.iesa.ufg.br/lapig/).

- ***2-*** Vegetação do Brasil x Desmatamento do Cerrado x Municípios do Brasil (***VDM***, 36470 resultados):
  - $DS(V)\, DS(D)\, JO_{(D)}\, DS(M)\, JO_{(M)}$;
- ***3-*** Municípios do Brasil x Localidades do Brasil x Rodovias do Brasil (***MLR***, 507556 resultados):
  - $DS(M)\, DS(L)\, JO_{(M)}\, DS(R)\, JO_{(R)}$.

4.1. Metodologia

Foram definidas 4 configurações de *cluster* para a realização dos experimentos: ***cluster1*** com 1 servidor, ***cluster2*** com 2 servidores, ***cluster4*** com 4 servidores e ***cluster8*** com 8 servidores. Cada servidor é uma instância de máquina virtual (MV) com 4 vCPUs de 2.4 Ghz, 8 GB de memória RAM, 12 GB de disco e com interface de rede 1 Gbit, com o SO Ubuntu 12.04 LTS.

Foram realizadas centenas de consulta e utilizadas na avaliação um total de 140 resultados de consultas de *multi-way spatial join* sobre diferentes configurações do sistema (i.e., combinações de Plano de Consulta, distribuição de dados, número de servidores e aplicação ou não do DSR). A fim de determinar qual o algoritmo de distribuição de dados se adequa melhor ao DST, o plano de consulta ***VDM*** foi processado sobre todas as configurações de *cluster*, utilizando as 2 distribuições de dados avaliadas, totalizando 40 consultas (Seção 4.2).

Para avaliar o desempenho e o grau de escalabilidade do DST os planos de consulta ***IHRF***, ***VDM*** e ***MLR*** foram processados sobre as 4 configurações de *cluster* somando mais 60 execuções ao total de consultas realizadas (5 execuções x 3 planos de consulta x 4 configurações de *cluster*). Os resultados são apresentados e discutidos na Seção 4.2.

Em cada uma dessas execuções foram coletadas as seguintes métricas: utilização média de CPU; média de kbytes trafegados na rede; quantidade de mensagens trocadas; tempo de resposta; quantidade de operações de junção binária; quantidade de planos de execução processados na fase de filtragem; quantidade de candidatos à solução da fase de filtragem e quantidade de falsos positivos na fase de refinamento.

Para viabilizar a experimentação do algoritmo DST foi utilizada a plataforma de middleware de geoprocessamento distribuído *DistGeo* [de Oliveira et al. 2013] que conta com uma arquitetura de comunicação baseada no modelo *peer-to-peer*, onde os servidores do cluster trocam informações de descoberta utilizando o protocolo *Gossip* [Demers et al. 1987], além de recursos interessantes como diferentes formas de distribuição de dados e o tratamento de concorrência, consistência no processamento de operações espaciais.

Cada uma das consultas avaliadas nos experimentos também foram processada em uma instância do PostGIS com o propósito de validar os resultados obtidos pelo DST que retornou os mesmos resultados que o PostGIS conforme esperado.

4.2. Avaliação de desempenho do algoritmo DST

Foram realizados experimentos utilizando os dois algoritmos de distribuição de dados *Round-robin* [de Oliveira et al. 2011] e *Grid-proximity* [de Oliveira 2013], a fim de selecionar aquele que melhor atende às particularidades do DST.

A distribuição *Grid-proximity-area* alcança o seu principal objetivo que é reduzir o tráfego de dados na rede. Por outro lado, ele gera gargalos de processamento devido à aglomeração de dados espacialmente próximos em um pequeno número de servidores. Com a utilização do *Round-robin*, graças à distribuição natural dos dados entre os servidores do *cluster*, apesar da geração de tráfego de dados um pouco maior (em alguns casos chegou a ser 9 vezes maior), a melhor distribuição dos dados e carga de processamento levou a um melhor aproveitamento dos recursos do *cluster*.

Em função do comportamento observados na análise dos algoritmos de distribuição, os demais resultados de avaliação de desempenho apresentados nessa seção foram obtidos com a utilização da distribuição de dados *Round-robin*, que se mostrou mais adequada ao algoritmo DST.

4.2.1. Plano de Consulta *1- IHRF*

A consulta IHRF é computacionalmente cara, devido ao grande número de operações de junção espacial binária (202.536.882 junções) e pela alta porcentagem de falsos positivos na fase de refinamento, aproximadamente 81% dos candidatos a solução. Isso se deve pela grande quantidade de espaço morto gerado pelos MBRs dos índices *R-Tree* das tabelas com geometrias do tipo Linha. Apesar do alto custo computacional dessa consulta, o DST conseguiu alcançar um bom grau de escalabilidade.

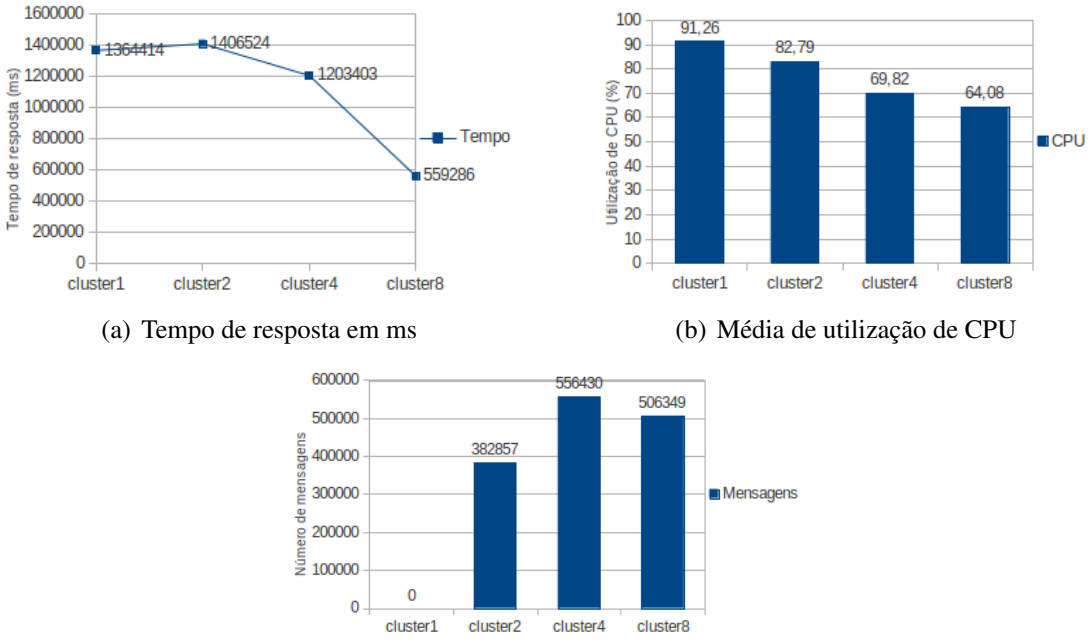
Entre os *clusters* com 1 e 2 servidores houve uma pequena piora de aproximadamente 3% no tempo de resposta, devido a introdução do tráfego de dados na rede (que não ocorre no ***cluster1***). A partir do cluster2 o tempo de resposta melhorou aproximadamente 16,9% no cluster4 e aproximadamente 115,17% no cluster8. Isto indica uma melhora progressiva no tempo de resposta a partir do ***cluster2*** conforme mostrado pela Figura 7(a).

O principal fator que contribuiu com o bom resultado obtido na consulta ***IHRF*** sobre o ***cluster8*** foi o alto grau de colocalização dos dados, evidenciada pela Figura 7(c). Essa colocalização foi gerada ao acaso, uma vez que o algoritmo de distribuição *Round-robin* não tem nenhum controle sobre a colocalização dos dados. Entretanto, este fenômeno favoreceu o desempenho do DST que não precisa trocar mensagens na rede para processar os dados colocalizados.

Por ter sido reduzido o tráfego de dados na rede, foi possível obter, em todas as configurações de *cluster*, uma porcentagem alta e uniforme de utilização de CPU, como pode ser observado na Figura 7(b). Esta alta porcentagem de utilização foi a responsável pelos fatores de escalabilidade horizontal obtidos.

4.2.2. Plano de Consulta *2- VDM*

A Figura 8(a) apresenta a escalabilidade do tempo de resposta da consulta ***VDM*** sobre as diferentes configurações de *cluster*. A Figura 8(a) mostra uma redução de aproximadamente 6,5% no tempo de resposta do ***cluster1*** para o ***cluster2***. Essa porcentagem de melhora discreta se deve à comunicação dos servidores pela rede no ***cluster2***, que provocou a troca de 29393 mensagens no **cluster2** (vide Figura 8(b)). No entanto, a partir do



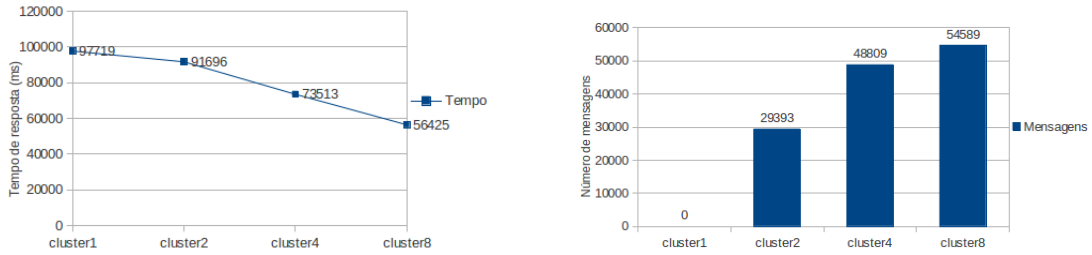
(a) Tempo de resposta em ms

(b) Média de utilização de CPU

(c) Número de mensagens trocadas

Figura 7. Consulta 1- Hidrovias X Hidrografia X Rodovias X Ferrovias.

cluster2, o fator de escalabilidade no tempo de resposta foi mais significativo, apresentando uma redução no tempo de resposta de 19,83% no ***cluster4*** em relação ao ***cluster2***, e de 23,24% no ***cluster8*** em relação ao ***cluster4***.



(a) Tempo de resposta em ms

(b) Número de mensagens trocadas

Figura 8. Consulta 2- Vegetação x Desmatamento x Municípios.

Conforme pode ser observado na Figura 8(b), o ***cluster4*** apresentou um aumento de aproximadamente 1,6 vezes no número de mensagens em relação ao ***cluster2***, no caso do ***cluster8*** esse aumento foi menor, 1,1 vezes em relação ao ***cluster4*** apesar do número de servidores ter dobrado.

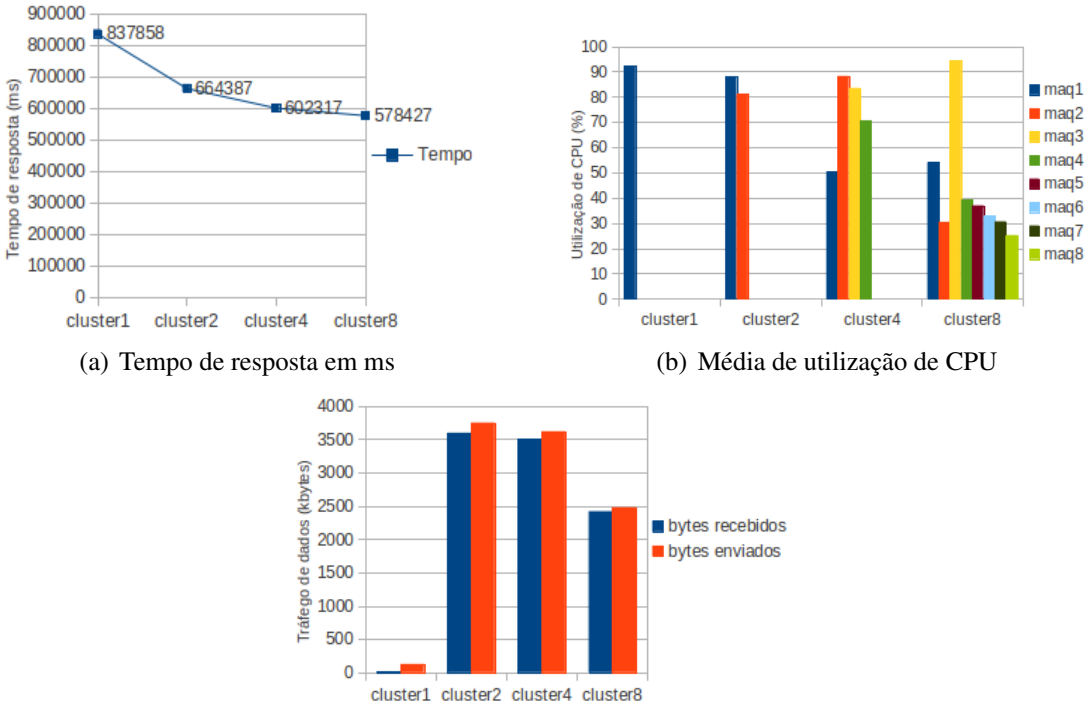
Isso indica que houve colocalização de dados das diferentes bases de dados envolvidas na consulta de forma balanceada sem a geração de gargalos de processamento (apesar dos dados espacialmente próximos estarem no mesmo servidor, os dados foram distribuídos entre os servidores de forma balanceada). Conforme discutido na Seção 4.2.1, essa colocalização foi gerada ao acaso. Isso provocou uma progressão menor no número de mensagens devido à redução da obtenção de dados na rede para o processamento da junção espacial das partes do plano de execução. Esse comportamento também ocorreu

no processamento da consulta ***IHRF***, só que em uma escala ainda maior (vide Seção 4.2.1).

A boa distribuição da carga de processamento fez com que a média de utilização de CPU em cada uma das configurações dos *clusters* reduzisse em uma taxa razoavelmente baixa. Isso garantiu uma porcentagem de utilização de CPU relativamente alta, o que possibilitou o fator de escala apresentado.

4.2.3. Plano de Consulta *3- MLR*

Na consulta ***3- MLR***, o DST também apresentou escala no tempo de resposta, mas foi observado um padrão de escala menor do que nas demais consultas, conforme ilustrado na Figura 9(a). Ao contrário do que ocorreu nas consultas ***IHRF*** e ***VDM***, o fator de escala piora conforme se acrescenta servidores no sistema. Com uma melhora de aproximadamente 20,7% do cluster1 para o cluster2, a partir do cluster2, a porcentagem de melhora reduz, caindo para 9,34% e aproximadamente 4% nas transições para *clusters* de 4 e 8 servidores.



(a) Tempo de resposta em ms

(b) Média de utilização de CPU

(c) Tráfego de dados na rede em kbytes

Figura 9. Consulta 3- Municípios x Localidades x Rodovias.

Este comportamento ocorreu porque com essa combinação houve uma tendência de formação de gargalos de processamento (i.e., poucas máquinas ficaram muito ocupadas enquanto outras ficaram ociosas). Devido à colocalização de dados em um pequeno número de servidores. A Figura 9(b) evidencia essa formação de gargalo, muito clara no ***cluster8***, onde a maioria dos servidores, teve média de uso de CPU abaixo de 30%, enquanto a média de um dos servidores foi de 90%.

A Figura 9(c) dá outra evidência ao mostrar a redução da média de dados enviados e recebidos na rede conforme se aumenta o número de servidores. Isso ocorre em parte pela colocalização dos dados, que acabou gerando o gargalo de processamento, e em parte pela característica das bases de dados, uma vez que dados geográficos do tipo ponto e linhas geram menos tráfego de dados do que polígonos [de Oliveira 2013].

5. Trabalhos Correlatos

Apesar da importância das consultas multi-way spatial join, poucos trabalhos investigaram o seu processamento em ambientes distribuídos, conforme pode-se observar na revisão da literatura feita em [Jacox and Samet 2007] e também em revisões literárias de trabalhos mais recentes como [Gupta et al. 2013]. Nosso trabalho apresenta um algoritmo distribuído para processamento de consultas multi-way spatial join, usando índices espaciais distribuídos e sem persistir resultados intermediários.

Gupta et. al. [Gupta et al. 2013] também apresenta um algoritmo para processamento distribuído de consultas multi-way spatial join. O algoritmo implementado é baseado em spatial hash join [Lo and Ravishankar 1996], que processa bases de dados não indexadas. O paradigma de programação distribuída map-reduce é empregado para o processamento distribuído da consulta. O espaço geográfico representado nas bases de dados é dividido em partes iguais utilizando uma grade, de forma que cada célula contenha uma quantidade de objetos espaciais. Cada célula é então mapeada na função map do paradigma e as células colocalizadas são correlacionadas para produzir os resultados da consulta na função reduce. Por natureza, o paradigma map-reduce persiste resultados intermediários em um sistema de arquivos distribuídos.

As diferentes abordagens entre nosso trabalho e o de [Gupta et al. 2013] tornam difícil uma comparação de tempo de processamento e uso de recursos. Porém, os tempos de processamento estão em grandezas diferentes (minutos contra horas). Apesar da diferença no tamanho das bases de dados processadas, há ainda um outro fator importante: nossos tempos incluem a etapa de refinamento, ou seja, o processamento dos objetos espaciais completos, enquanto [Gupta et al. 2013] afirmou considerar apenas uma simplificação dos objetos espaciais (MBR). A etapa de refinamento é a etapa mais demorada de uma junção espacial, conforme discutido em [Jacox and Samet 2007].

Outras abordagens usando o paradigma map-reduce foram publicadas recentemente, como [Zhang et al. 2009], e propõem técnicas para o processamento de consultas de junção espacial binária, utilizando algoritmos baseados em spatial hash-join, sem indexar as bases de dados. [de Oliveira et al. 2013] descreve um algoritmo de processamento distribuído da junção espacial binária, porém utilizando o paradigma peer-to-peer e empregando bases de dados indexadas. Todas estas técnicas, por si só, não são suficientes para o processamentos de consultas multi-way spatial join. É necessário definir como os resultados intermediários serão combinados em etapas sucessivas.

Nesta direção, [Mamoulis and Papadias 2001] apresenta um estudo extensivo de um otimizador de consultas, o qual define um plano de execução em etapas sucessivas, com uma combinação de algoritmos de junção espacial. Os autores usam histogramas de cardinalidade com duas dimensões para estimar a seletividade das junções e reduzir os resultados intermediários, evitando assim planos de execução ruins. No entanto, somente o uso de algoritmos de junção centralizados e o custo de E/S local são considerados na

definição dos planos. Seus resultados não podem ser diretamente aplicados aos algoritmos de junção distribuídos sem que antes seja estudado o impacto da distribuição dos dados no tempo de execução final da consulta.

6. Conclusões

Neste trabalho, buscou-se explorar o potencial dos sistemas distribuídos na solução de um problema complexo e com alto custo computacional, o processamento da junção espacial de um número arbitrário de bases de dados de entrada (*multi-way spatial join*).

Para tanto, muitos desafios tiveram que ser superados, a definição de um Plano de Execução que possibilitasse mapear a semântica das consultas de *multi-way spatial join* e, ao mesmo tempo, coordenar de forma distribuída o processamento das múltiplas operações de junção espacial.

O maior desafio em qualquer sistema distribuído é distribuir as tarefas o mais uniformemente possível para evitar subutilização de recursos computacionais. Os resultados demonstraram que o algoritmo DST conseguiu superar este desafio, ao se mostrar escalável em todos os experimentos realizados.

O algoritmo DST apresentou um bom grau de escalabilidade em consultas formadas por diferentes tipos de bases de dados reais, mesmo ao serem processadas através de Planos de Consulta simples, que não levam em conta na sua construção fatores como tipo ou cardinalidade das bases de dados.

Durante a confecção deste trabalho, foram identificados diversos pontos de otimização que podem melhorar o desempenho do DST. Os principais deles foram elencados abaixo:

- Definir algoritmos de otimização para a conversão automática de grafos de consulta em planos de consulta;
- Melhorar o algoritmo para paralelizar o processamento (em um mesmo nível) de partes independentes do Plano de Execução;
- Desenvolver um algoritmo de distribuição de dados mais adequado ao *multi-way spatial join*.

Referências

Bacchus, F. and van Beek, P. (1998). On the conversion between non-binary and binary constraint satisfaction problems. In *AAAI/IAAI*, pages 310–318.

Beckmann, N., Kriegel, H., Schneider, R., and Seeger, B. (1990). *The R*-tree: an efficient and robust access method for points and rectangles*, volume 19. ACM.

Bentley, J. (1975). Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):517.

Brinkhoff, T., Kriegel, H., and Seeger, B. (1993). *Efficient processing of spatial joins using R-trees*, volume 22. ACM.

Comer, D. (1979). Ubiquitous B-tree. *ACM Computing Surveys (CSUR)*, 11(2):121–137.

de Oliveira, S., Sacramento, V., Cunha, A., Aleixo, E., de Oliveira, T. B., de Castro Cardoso, M., and Junior, R. R. (2013). Processamento Distribuído de Operações de Junção

Espacial com Bases de Dados Dinâmicas para Análise de Informações Geográficas. In *SBRC 2013 (XXXI Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos)*, Brasília, Brazil.

de Oliveira, S. S. T. (2013). DistJoin: Plataforma de Processamento Distribuído de Operações de Junção Espacial com Bases de Dados Dinâmicas. Mestrado, Instituto de Informática - Universidade Federal de Goiás.

de Oliveira, T., Sacramento, V., Oliveira, S., Albuquerque, P., and Cardoso, M. (2011). DSI-RTree-Um Índice R-Tree Escalável Distribuído. In *SBRC 2011 (XXIX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos)*, Campo Grande, Brazil.

Demers, A., Greene, D., Hauser, C., Irish, W., Larson, J., Shenker, S., Sturgis, H., Swinehart, D., and Terry, D. (1987). Epidemic algorithms for replicated database maintenance. In *Proceedings of the sixth annual ACM Symposium on Principles of distributed computing*, pages 1–12. ACM.

Gupta, H., Chawda, B., Negi, S., Faruquie, T. A., Subramaniam, L. V., and Mohania, M. (2013). Processing multi-way spatial joins on map-reduce. In *Proceedings of the 16th International Conference on Extending Database Technology*, pages 113–124. ACM.

Guttman, A. (1984). *R-trees: a dynamic index structure for spatial searching*, volume 14. ACM.

Jacox, E. H. and Samet, H. (2007). Spatial join techniques. *ACM Transactions on Database Systems (TODS)*, 32(1):7.

Kamel, I. and Faloutsos, C. (1994). Hilbert R-tree: An Improved R-tree using Fractals. In *VLDB 20th*, page 509. Morgan Kaufmann Publishers Inc.

Lo, M.-L. and Ravishankar, C. V. (1996). Spatial hash-joins. In *ACM SIGMOD Record*, volume 25, pages 247–258. ACM.

Mamoulis, N. and Papadias, D. (2001). Multiway spatial joins. *ACM Transactions on Database Systems (TODS)*, 26(4):424–475.

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.

Papadias, D., Mamoulis, N., and Delis, V. (1998). Algorithms for querying by spatial structure. In *Proceedings of the 24rd International Conference on Very Large Data Bases*, pages 546–557. Morgan Kaufmann Publishers Inc.

Papadias, D., Mamoulis, N., and Theodoridis, Y. (1999). Processing and optimization of multiway spatial joins using r-trees. In *Proceedings of the eighteenth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 44–55. ACM.

Prosser, P. (1993). Hybrid algorithms for the constraint satisfaction problem. *Computational intelligence*, 9(3):268–299.

Zhang, S., Han, J., Liu, Z., Wang, K., and Feng, S. (2009). Spatial queries evaluation with mapreduce. In *Grid and Cooperative Computing, 2009. GCC'09. Eighth International Conference on*, pages 287–292. IEEE.