Processamento Distribuído de Operações de Junção Espacial com Bases de Dados Dinâmicas para Análise de Informações Geográficas

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

[1]Instituto de Informática – Universidade Federal de Goiás (UFG)
Bloco IMF I, Campus II - Samambaia – Goiânia – GO – Brasil

{savio.teles,vagner.sacramento,anderson.cunha,everton.lima,
thiago.borges,marcelo.cardoso,roberto.junior}@lupa.inf.ufg.br

Abstract. *This paper presents the proposal of a new data distribution technique, called Proximity Area, that optimizes the processing of distributed spatial join operations for analysis of a large volume of spatial data in dynamic datasets. The techniques found in the literature, for processing distributed spatial join, perform data distribution in static datasets, where it is necessary redistribute the objects on cluster for each dataset update. This becomes unfeasible for datasets with large volume of data and constant updates. The experiments have shown the efficiency of the Proximity Area and the impact of the data distribution on distributed spatial join.*

Resumo. *Este artigo apresenta a proposta de uma nova técnica de distribuição de dados, denominada Proximity Area, que otimiza o processamento de operações de junção espacial distribuída para análise de um grande volume de dados geográficos em bases de dados dinâmicas. As técnicas encontradas na literatura, para processamento da junção espacial, realizam a distribuição de dados em bases de dados estáticas, onde é necessário redistribuir os objetos pelo cluster a cada atualização da base de dados. Isto se torna inviável para bases de dados com grande volume de dados e com constantes atualizações. Os experimentos realizados demonstraram a eficiência da Proximity Area e apresentaram o impacto da distribuição de dados sobre a junção espacial distribuída.*

1. Introdução

Se a resposta para a pergunta "Onde?" é importante para o seu problema, um Sistema de Informação Geográfico (SIG) é a solução. No entanto, só visualizar dados em um mapa não supre a demanda de informações para tomada de decisão que precisam fazer análises de grandes volumes de dados em tempo real. Em muitos casos, é necessário realizar uma operação de junção espacial, correlacionando diferentes camadas de dados. Por exemplo: encontrar as áreas desmatadas no Brasil que estão próximas de leitos de rios.

Os algoritmos de processamento da junção espacial apresentam alto custo computacional [Mutenda and Kitsuregawa 1999]. Em função disto, as soluções de análise de dados geográfico que manipulam grande volume de dados geográficos devem ser capazes

de paralelizar o processamento da junção espacial entre os computadores de um *cluster*. Com isso, alguns desafios são identificados: i) distribuição dos dados pelo *cluster* e ii) processamento paralelo e distribuído da junção espacial.

O objetivo deste trabalho é apresentar uma nova técnica de distribuição de dados, denominada *Proximity Area*, que otimiza o processamento de operações de junção espacial distribuída para análise de grande volume de dados geográficos em bases de dados dinâmicas. Os trabalhos na literatura têm explorado a distribuição em bases de dados estáticas, onde qualquer atualização da base de dados requer que todos os dados sejam novamente distribuídos pelo *cluster*. Isto se torna inviável em bases de dados com grandes volumes de dados e que sofrem constantes atualizações. *Proximity Area* consegue atualizar a base de dados sem que haja a necessidade de redistribuir os objetos pelo *cluster*. Esta técnica foi implementada sobre a plataforma de *middleware* para geoprocessamento distribuído, DistGeo, baseada no modelo peer-to-peer para comunicação entre os servidores.

As principais contribuições deste trabalho são:

- Implementação de um algoritmo de junção espacial distribuída para análise de dados geográficos;
- Uma nova técnica de distribuição de dados para bases de dados dinâmicas.

O restante do trabalho está organizado como segue. A Seção 2 apresenta uma revisão das estruturas de dados espaciais existentes e a visão geral da operação de junção espacial. A Seção 3 apresenta a arquitetura da plataforma DistGeo e a técnica de distribuição de dados *Proximity Area*. A Seção 4 descreve a metodologia e os resultados dos testes executados na plataforma. A Seção 5 compara a plataforma DistGeo com as estratégias encontradas na literatura para processar a junção espacial distribuída. A Seção 6 apresenta as conclusões deste trabalho e uma breve descrição dos trabalhos futuros.

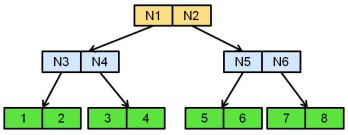
2. Processamento de Dados Espaciais

A indexação de dados espaciais vetoriais vem sendo pesquisada desde 1975, o que ocasionou o surgimento de diversas estruturas de dados, sendo as principais: KD-Tree [Bentley 1975], Hilbert R-Tree [Kamel and Faloutsos 1994] e a R-Tree [Guttman 1984].

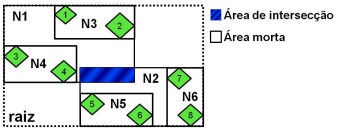
A R-Tree é uma árvore balanceada por altura, semelhante a B^+-Tree, com ponteiros para objetos espaciais nos nós folhas. É uma estrutura de dados hierárquica que utiliza retângulos para organizar um conjunto dinâmico de objetos espaciais, de maneira que objetos co-localizados fiquem armazenados próximos uns dos outros e que haja uma redução no espaço de busca a cada nível da árvore [Guttman 1984]. Estes retângulos, chamados de MBRs, *Minimum Bounding Rectangle*, possuem área menor possível para envolver as geometrias dos filhos. Na Figura 1(b), o MBR de N3 é o menor possível para envolver os filhos 1 e 2.

A Figura 1(a) ilustra a estrutura hierárquica da R-Tree com um nó raiz, nós internos (N1..2 $\subset$ N3..6) e um último nível de nós folha (N3..6 $\subset 1..8$). Cada nó armazena no máximo M e no mínimo $m \leq \frac{M}{2}$ entradas [Guttman 1984]. A Figura 1(b) retrata o desenho dos MBRs agrupando os objetos espaciais de 1 a 8 em subconjuntos, de acordo com sua co-localização.

Entre as várias extensões propostas para a R-Tree, a R*-Tree [An et al. 2003] foi escolhida para implementação da arquitetura proposta neste trabalho. Esta variante propõe mecanismos para melhorar o tempo de busca [Beckmann et al. 1990] como reduzir espaço morto e áreas sobrepostas entre os MBRs. Espaço morto é a área adicional do MBR necessária para cobrir o polígono como um todo. Na Figura 1(b), a área de N1 não preenchida pelas suas entradas é um exemplo de espaço morto. Áreas sobrepostas são regiões de interseção entre polígonos. Um exemplo de sobreposição é ilustrado na Figura 1(b), entre N1 e N2.



(a) Índice espacial *R-Tree*

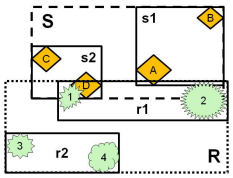


(b) Disposição espacial dos objetos geográficos

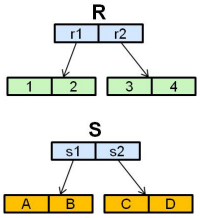
Figura 1. Estrutura de dados espacial

2.1. Junção espacial

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, $r_i \bigcap s_j \neq \emptyset$ [Jacox and Samet 2007]. Este trabalho utiliza um algoritmo de junção espacial que caminha por duas relações R e S indexadas por *R*-Trees* [Brinkhoff et al. 1993].



(a) Disposição espacial das relações R e S



(b) Indexação das relações R e S na *R*-Tree*

Figura 2. Junção espacial entre as relações R e S

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 utiliza aproximações das geometrias dos objetos [1] na operação de intersecção para gerar um conjunto de possíveis respostas à consulta. Observando a Figura 2(a), percebe-se que o MBR de r1 intersecta

[1] Aproximações são polígonos utilizados para reduzir a complexidade das geometrias para poucos pontos geométricos [Brinkhoff et al. 1994]. Por exemplo: uma geometria com milhares de pontos pode ser representada por um MBR com 2 pontos. Estas aproximações geram espaços mortos e, por isso, são utilizadas na operação de intersecção para descartar resultados incorretos. A intersecção entre duas aproximações não significa que suas respectivas geometrias também intersectam.

com os MBRs de s1 e s2, mas que o MBR de r2 não intersecta com nenhum item na relação S. Por isso, o conjunto de saída da etapa de filtragem é formado pelos pares (r1, s1) e (r1, s2). A fase de refinamento é realizada nas folhas e remove deste conjunto os resultados incorretos utilizando as geometrias reais de cada objeto. Observando a Figura 2(b), a etapa de refinamento irá analisar os nós filhos de (r1, s1) e (r1, s2). Nesta fase houve a necessidade de verificar os seguintes conjuntos de pares de candidatos {(1, A), (1, B), (1, C), (1, D), (2, A), (2, B), (2, C), (2, D)} e apenas (1, D) fez parte do resultado final por apresentar intersecção entre suas respectivas geometrias.

3. Processamento da junção espacial distribuída

Os algoritmos de junção espacial apresentam alto custo de processamento e, por isso, as pesquisas têm se concentrado em resolver o problema de forma distribuída. A Figura 3 ilustra as duas *R*-Trees*, R e S, apresentadas na Figura 2(b) distribuídas em um *cluster* de computadores. Os algoritmos de inserção e junção espacial executados na *R*-Tree* centralizada podem ser processados de forma semelhante na versão distribuída, exceto pela i) necessidade de troca de mensagens para acessar os objetos distribuídos e ii) pelo tratamento de concorrência e consistência necessário para ao processamento paralelo e distribuído dos algoritmos [de Oliveira et al. 2011].

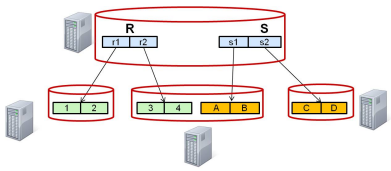


Figura 3. R-Tree distribuída

Nem sempre os dados necessários para a junção entre as bases de dados estão disponíveis localmente. No exemplo da Seção 2.1, a junção espacial entre R e S teve como resultado o par (1, D). Observando a Figura 3, percebe-se que os objetos 1 e D estão localizados em máquinas diferentes. Portanto, para processar a junção espacial entre os dois objetos, um deles deve ser trafegado na rede até o local em que está armazenado o outro objeto. Para reduzir o tráfego de dados na rede, o algoritmo de processamento da junção espacial da plataforma DistGeo trafega o objeto com menor número de pontos.

3.1. DistGeo: Plataforma de Geoprocessamento Distribuído de Operações Espaciais

Como pode ser visto na Figura 4, a plataforma DistGeo é constituída pelas aplicações clientes, servidores que executam as operações espaciais e um serviço de nomes replicado. O serviço de nomes armazena informações sobre os servidores ativos e as bases de dados que foram inseridas no sistema. As aplicações clientes se comunicam com a plataforma através de uma API cliente, que disponibiliza métodos para atualização e consultas nas bases de dados. O serviço de nomes é consultado pela API cliente para decidir em qual servidor executar a operação espacial desejada. A API cliente envia uma requisição e recebe uma resposta do um servidor escolhido e repassa o resultado da operação para a aplicação cliente. Cada servidor é responsável por executar as operações espaciais requisitadas pelas aplicações clientes.

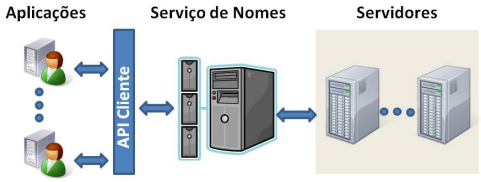


Figura 4. Plataforma DistGeo

A arquitetura da plataforma DistGeo é baseada no modelo *peer-to-peer* híbrido, na qual os servidores trocam mensagens entre si, mas necessitam do serviço de nomes para obterem o endereço IP dos outros servidores. Cada servidor guarda em uma *cache* os endereços já obtidos, para que não seja sempre necessário acessar o serviço de nomes para se comunicar com outro servidor. A plataforma possui tratamentos de concorrência e consistência semelhantes aos apresentados em [de Oliveira et al. 2011].

O primeiro servidor que inicia no *cluster* se cadastra no serviço de nomes como monitor. Este servidor tem ciência de quais objetos foram inseridos e em quais máquinas foram armazenados. Para cada objeto novo no sistema, deve-se requisitar as informações do *cluster* ao servidor monitor para decidir para qual máquina enviar o novo objeto. Depois de alocado, deve-se informar ao monitor em qual servidor este objeto foi armazenado. O serviço de nomes monitora cada servidor e, quando o monitor cai, ele fica ciente e avisa aos outros servidores do *cluster*. Neste caso, os outros servidores tentam se cadastrar como monitor e o primeiro que conseguir, assume a função utilizando as últimas informações obtidas do antigo monitor.

3.2. Estratégia de distribuição em bases de dados dinâmica

A distribuição de dados pelas máquinas do *cluster* é o fator que mais influencia no paralelismo em um ambiente *clusterizado* [Mutenda and Kitsuregawa 1999] e possui dois requisitos principais [Patel and DeWitt 2000]: a) os dados devem ser distribuídos de forma balanceada pelas máquinas do *cluster* e b) uma máquina deve possuir a maior parte dos dados que precisa para processar uma operação localmente, ou seja, não é necessário obter dados de outra máquina.

Este trabalho apresenta um nova técnica de distribuição para bases de dados dinâmicas denominada *Proximity Area*. Esta busca manter objetos, de bases de dados diferentes, espacialmente próximos na mesma máquina (**co-localizar**) para que ocorra menos tráfego na rede nas operações de junção espacial. Para que os dados fiquem distribuídos de forma balanceada, foi criado um fator de balanceamento **k** - entre 0 e 1 - que limita a diferença na quantidade de objetos entre os servidores. Manter o *cluster* balanceado aumenta o grau de paralelismo na execução da operação de junção espacial distribuída e, consequentemente, é possível aproveitar melhor os recursos disponíveis no *cluster*.

O Algoritmo 1 apresenta a descrição da técnica *Proximity Area*. Este algoritmo tem como objetivo escolher o servidor S no qual o novo objeto O será alocado e recebe três parâmetros: a) MBR de O, b) fator de balanceamento k e c) $lista$, obtida no monitor, que contém as informações dos servidores do *cluster*. O servidor monitor armazena duas informações de cada máquina: i) o MBR que engloba os objetos de todas as bases de dados naquela máquina; ii) quantidade de objetos. Cada elemento I em $lista$ possui

como atributos a referência para o servidor S correspondente a I, o número de objetos em S e o MBR que representa a área espacial que engloba todos os objetos das diferentes bases de dados alocados em S.

Algoritmo 1: $ProximityArea(M, k, lista)$

Entrada: M MBR do objeto alocado, k fator de balanceamento, $lista$ informações de distribuição dos servidores do cluster
Saída: Referência para o servidor escolhido

```
1  min ← inteiro máximo
2  para cada elemento I em lista faça
3      se I.numObjetos = 0 então
4          retorna I.referenciaServidor
5      fim
6      se I.numObjetos < min então
7          min ← I.numObjetos
8      fim
9  fim
10 minArea ← número de ponto flutuante máximo
11 referenciaServidor ← null
12 para cada elemento I em lista faça
13     se (min/I.numObjetos) > k então
14         area ← aumento de área de I.MBR para inserir M
15         se area < minArea então
16             minArea ← area
17             referenciaServidor ← I.referenciaServidor
18         fim
19     fim
20
21 fim
22 retorna referenciaServidor
```

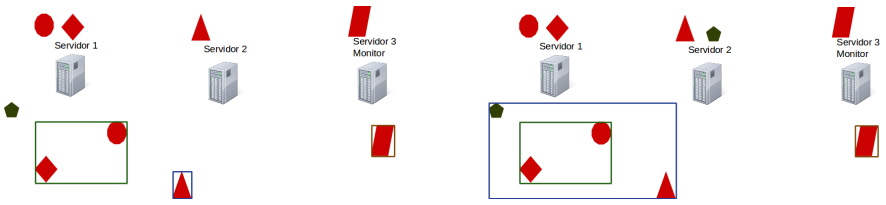
O Algoritmo 1 realiza um primeiro laço entre as linhas 2 e 9 que verifica se existe algum servidor que não tenha nenhum objeto alocado. Neste caso, o algoritmo retorna este servidor como resposta. Caso não exista nenhum servidor vazio, ao final do laço a variável min guardará o número de objetos do servidor com menor quantidade de dados alocados.

No segundo laço, na linha 13 é verificado se o servidor S em questão contém um número de objetos que não obedecem ao fator de balanceamento k. Para isso, a divisão entre min e o número de objetos $I.numObjetos$ em S deve ser maior que k. Se, por exemplo, $min = 1$, $k = 0,5$ e $I.numObjetos = 2$, então $(min/I.numObjetos) = 0,5$ não é maior que k e, neste caso, devemos descartar este servidor na alocação de O. Entre os servidores que obedecerem o fator de balanceamento, é escolhido aquele cujo MBR está mais próximo espacialmente do MBR do novo objeto O.

Objetos espaciais têm características de distribuição não uniforme e, como não é possível controlar a ordem com que os objetos espaciais de uma base de dados são inseridos, as técnicas de distribuição dinâmicas de dados não conseguem alocar estes

objetos da melhor forma possível [Zhou et al. 2011]. A técnica *Proximity Area* também apresenta este comportamento, como pode ser visto na Figura 5 na inserção de um novo objeto (pentágono) com $k = 0,5$. O objeto deveria ser alocado no Servidor 1, que possui área espacial mais próxima do pentágono. Mas, como $min = 1$ e o Servidor 1 possui 2 objetos, então o número de objetos no Servidor 1 não obedece o fator de balanceamento. Com isto, o objeto será alocado no Servidor 2 que obedece o fator de balanceamento e está mais próximo espacialmente do objeto que o Servidor 3.

Assim, observando a Figura 5, percebe-se que a área do Servidor 2 não contém objetos co-localizados e, isto, gera um aumento no número de mensagens na rede na operação de junção espacial distribuída. Este problema será mitigado em trabalhos futuros, através da redistribuição dos dados entre os servidores em determinados intervalos de tempo.



(a) Antes da inserção (b) Depois da inserção

Figura 5. Inserção de um objeto utilizando a técnica *Proximity Area* com k = 0,5

4. Avaliação de Performance

Para analisar o desempenho da plataforma com a técnica de distribuição de dados *Proximity Area*, foram medidos o tempo de resposta e a quantidade de bytes transferidos na rede para realizar a operação de junção espacial. A quantidade de bytes transferidos na rede é importante para verificar o quanto os dados estavam co-localizados. Quanto mais co-localizados, menor é o tráfego de rede, mas em contrapartida, a distribuição do processamento pelas máquinas do *cluster* fica prejudicada.

4.1. Ambiente Experimental e Bases de Dados

Foram utilizadas bases de dados geográficas disponibilizadas pelo LAPIG (Laboratório de Processamento de Imagens e Geoprocessamento)[2], que permitiram avaliar a plataforma DistGeo em um ambiente de *cluster* com dados geográficos reais. Para estudar o impacto das peculiaridades de cada base de dados na performance da plataforma, foram avaliadas bases de dados com características diferentes em relação a: a) Tipo de geometria: polígonos, linhas e pontos; b) Número de itens e c) Tamanho em disco. As bases de dados utilizadas estão descritas na Tabela 1.

Para avaliar o desempenho da plataforma na execução da junção espacial distribuída, foram realizadas as seguintes junções: a) Bioma do Cerrado e Desmatamento do Cerrado, que retorna 60798 resultados; b) Bioma da Caatinga e Localidades, que retorna 3934 resultados; c) Rodovias e Hidrografia, que retorna 55764 resultados. Essa combinação de experimentos permite avaliar a junção espacial com bases de dados com características diferentes, conforme pode ser observado na Tabela 1.

[2]www.lapig.iesa.ufg.br

Tabela 1. Descrição das bases de dados

Base de Dados	Geometria	Número de itens	Tamanho(MB)
Desmatamento do Cerrado	Polígono	32578	11,2
Bioma do Cerrado	Polígono	151986	411,3
Bioma da Caatinga	Polígono	10994	275,3
Hidrografia	Linha	226963	64,5
Rodovias	Linha	51645	15,2
Localidades	Ponto	21840	1,4

Foram encontradas na literatura duas técnicas de distribuição de dados para bases dinâmicas. Em [Zhou et al. 2011], os objetos espacialmente próximos são enviados para máquinas diferentes. Esta técnica é uma estratégia interessante para paralelizar consultas em apenas uma base de dados. Para a junção espacial se torna inviável, pois gera um aumento na quantidade de tráfego na rede por não manter os dados co-localizados. Em [de Oliveira et al. 2011] é apresentado uma técnica semelhante a Round Robin, onde os dados são distribuídos de forma balanceada pelo *cluster*. Esta técnica, entretanto, não foi testada em operações de junção espacial. Por isso, o desempenho da plataforma foi analisado utilizando duas técnicas de distribuição de dados: i) *Proximity Area* e ii) *Round-Robin*.

Para avaliar a técnica *Proximity Area*, foram utilizados como fator de balanceamento os valores 0.1 (PA 0.1), 0.5 (PA 0.5) e 0.9 (PA 0.9). Esta variação no fator de balanceamento permite investigar o impacto do balanceamento e a co-localização dos dados. Geralmente, quanto menor o fator de balanceamento mais co-localizados estão os dados, entretanto menos balanceado fica o *cluster*.

Os testes foram realizados com máquinas Optiplex 780 Intel Core 2 Quad 2.83GHz, com 4 Gb de memória RAM e foram configurados da seguinte forma: doze Servidores, um Cliente e um Servidor de Nomes. As máquinas foram conectadas por uma rede Ethernet 1Gbit/segundo e um switch Dell PowerConnect 6248P.

A junção espacial distribuída foi avaliada com 1, 4, 6, 8, 10 e 12 servidores. Cada junção descrita anteriormente foi realizada cinco vezes, em cada configuração do *cluster*, onde foi descartada a junção de maior tempo de execução e a de menor tempo e calculada uma média aritmética dos tempos das três restantes.

4.2. Avaliação

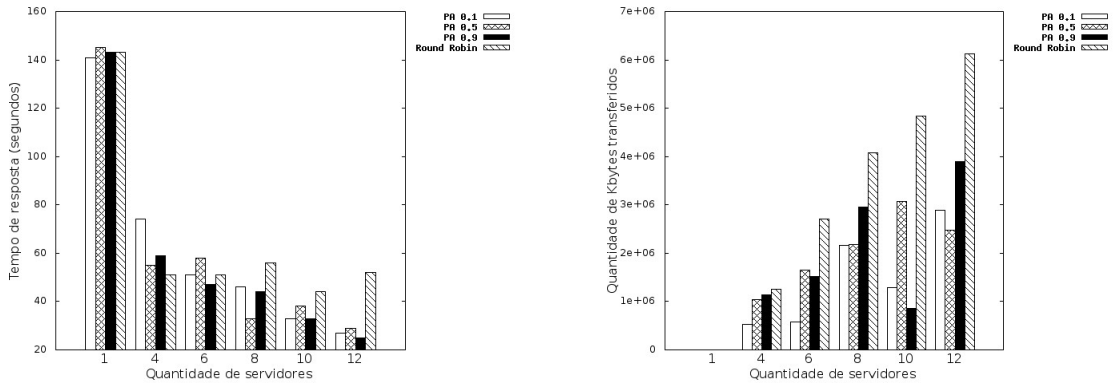
A Figura 6 apresenta o resultado das junções espaciais realizadas entre Desmatamento do Cerrado e Biomas do Cerrado. Estas duas bases de dados possuem geometrias complexas (grande número de pontos) e, por isso, a junção espacial apresenta um alto custo de processamento e tráfego de dados na rede. A Figura 6(a) demonstra que a plataforma DistGeo apresentou escalabilidade com o acréscimo de máquinas no *cluster* devido a alta demanda de processamento desta junção.

Como pode ser visto na Figura 6(b), a diferença na quantidade de bytes transferida na rede entre *Round Robin* e as outras técnicas é grande, pois *Round Robin* não consegue manter os itens co-localizados. Por isso, a técnica *Proximity Area* apresentou um desempenho melhor que *Round Robin* com o acréscimo de máquinas, pois a diferença no tráfego

de dados na rede aumenta consideravelmente.

Como a diferença de tráfego de dados na rede entre PA 0.9 e as técnicas PA 0.1 e PA 0.5 foi pequena, PA 0.9 apresentou melhor desempenho, pois consegue além de co-localizar os dados, distribuir o processamento entre as máquinas do *cluster*. A técnica PA 0.9 apresentou o melhor desempenho entre todas as técnicas, sendo, na média, 21% melhor que *Round Robin*.

Assim, em junções que apresentam um alto custo de processamento e tráfego de dados na rede, um fator de balanceamento alto consegue paralelizar o processamento e, ao mesmo tempo, manter uma co-localização satisfatória dos dados se comparado ao *Round Robin*.



(a) Tempo de resposta (b) Quantidade de Kbytes tranferidos na rede

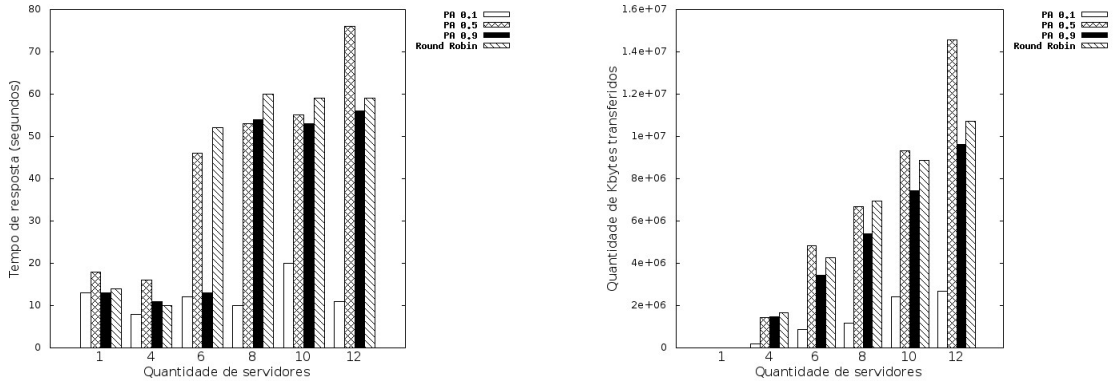
Figura 6. Junção espacial entre Bioma do Cerrado e Desmatamento

A junção entre as bases de dados Rodovias e Hidrografia analisa uma grande quantidade de pares de objetos que não fazem parte do resultado da consulta. Isso porque estas bases de dados possuem geometrias de linhas que geram MBRs com grande espaço morto. Este espaço morto exige que uma grande quantidade de pares de objetos das bases de dados sejam analisados para gerar o resultado da consulta. Para analisar estes pares, devem ser realizadas cópias de objetos entre as bases de dados para processar a junção espacial. Isto resulta em uma grande quantidade de tráfego de dados na rede.

Métricas coletadas nos testes mostraram que o uso da CPU no processamento desta junção é pequeno (em média 30%) porque as geometrias são simples e, com isto, o tráfego de dados se torna dominante. Por isso, pode ser observado na Figura 7 que o tempo de resposta da junção espacial foi proporcional ao tráfego de dados na rede. As técnicas não se beneficiaram com a adição de máquinas, já que a quantidade de bytes transferidos aumenta com um número maior de servidores - Figura 7(b). Como pode ser visto na Figura 7(a), a técnica *Proximity Area* apresentou na média desempenho melhor que *Round Robin*, pois consegue co-localizar os dados.

PA 0.1 teve o melhor desempenho (4 vezes melhor que *Round Robin*), pois consegue concentrar objetos espacialmente próximos na mesma máquina. Portanto, para junções onde o tráfego de dados na rede é dominante, um fator de balanceamento menor gera um melhor resultado, pois reduz o tráfego na rede através de uma melhor co-localização dos dados. Entre as restantes, PA 0.9 teve o melhor desempenho, sendo 29% melhor que a técnica *Round Robin*.

PA 0.5 apresentou um comportamento estranho neste teste, pois a quantidade de tráfego de dados trafegados na rede foi maior que as técnicas PA 0.9 e *Round Robin*, apesar de teoricamente PA 0.5 conseguir co-localizar melhor os dados. Por isso, PA 0.5 teve o pior desempenho entre todas as técnicas. Como foi dito na Seção 3.2, estas situações podem ocorrer devido a ordem de inserção dos itens e serão investigadas em trabalhos futuros.



(a) Tempo de resposta (b) Quantidade de Kbytes tranferidos na rede

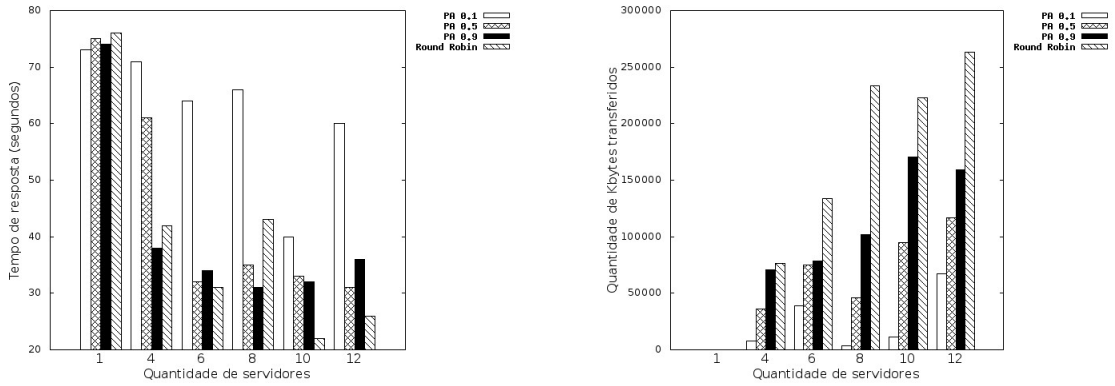
Figura 7. Junção espacial entre Hidrografia e Rodovias

Conforme apresentado na Seção 3, a plataforma DistGeo trafega na junção espacial os objetos com menor número de pontos para reduzir a quantidade de dados na rede. Na junção espacial entre Bioma da Caatinga e Localidades, a base de dados de Localidades é transferida, já que apresenta geometrias de pontos, enquanto Bioma da Caatinga apresenta geometrias complexas. Por isso, pode ser observado na Figura 8(b) que esta junção apresentou uma baixa quantidade de dados transferidos na rede, já que a base de dados Localidades possui pouco tamanho em disco (Tabela 1).

Métricas coletadas no testes mostraram que o uso da CPU no processamento desta junção não foi pequeno (em média 60%), pois a base de dados de Bioma da Caatinga apresenta geometrias muito complexas e, que requerem considerável uso da CPU mesmo em uma junção com uma base de dados com geometrias de Ponto (neste caso Localidades). Como o tráfego de dados na rede foi baixo, o tempo de processamento tornou-se dominante sobre o tráfego de dados na rede.

PA 0.9 e *Round Robin* apresentaram os melhores desempenhos entre todas as técnicas, pois conseguem distribuir o processamento da junção espacial entre as máquinas do *cluster*. PA 0.9 teve desempenho 3% pior que a técnica *Round Robin*. A técnica PA 0.1 apresentou o pior desempenho entre todas as técnicas, pois concentra o processamento em um pequeno conjunto de máquinas, subutilizando os recursos disponíveis no *cluster*. Portanto, para junções onde o processamento é dominante, um fator de balanceamento alto é melhor, pois balanceia os dados pelo *cluster* e, consequentemente, aumenta o paralelismo do processamento da junção espacial distribuída.

Os testes demonstraram que a técnica de distribuição de dados proposta neste trabalho, *Proximity Area*, apresentou um desempenho, no geral, melhor que *Round Robin*. Nos casos em que o trafego de dados na rede influenciaram de forma significativa no tempo de resposta, a diferença no tempo de resposta foi maior, pois a técnica *Proximity*



(a) Tempo de resposta (b) Quantidade de Kbytes tranferidos na rede

Figura 8. Junção espacial entre Bioma da Caatinga e Localidades

Area consegue co-localizar os dados, reduzindo assim o tráfego de dados na rede. A técnica *Round Robin* teve um bom desempenho quando houve pouco tráfego de dados na rede, apresentando um tempo de resposta próximo a PA 0.9. No geral, PA 0.9 apresentou o melhor desempenho entre todas as técnicas analisadas sendo, na média entre todos os testes, 16% melhor que *Round Robin*.

5. Trabalhos Correlatos

[Koudas et al. 1996] apresenta a proposta de uma arquitetura, onde o índice fica armazenado um uma máquina - *master* - e os dados são distribuídos pelas outras máquinas disponíveis - clientes. Esta abordagem gera um gargalo na máquina que armazena o índice e [Schnitzer and Leutenegger 1999] tenta resolver esse problema criando índices nos clientes para os dados armazenados localmente e deixando no *master* um índice para os clientes. Esta abordagem de [Schnitzer and Leutenegger 1999] também gera um gargalo no *master* para uma grande quantidade de consultas. Em [An et al. 1999] é proposta uma arquitetura semelhante a [Koudas et al. 1996] e, por isso, apresenta os mesmos problemas descritos anteriormente. Para resolver o problema de gargalos no sistema, [Mutenda and Kitsuregawa 1999] propôs uma arquitetura, onde o índice é replicado entre todas as máquinas do *cluster* e os dados são distribuídos segundo uma política circular (*Round-Robin*). Entretanto, a replicação do índice gera grande redundância de dados.

[Tan et al. 2000] analisa o impacto da cardinalidade de R e S no processamento da junção espacial distribuída. [Ramirez and de Souza 2001] avalia o impacto de aproximações mais complexas no processamento da junção espacial que, geralmente, utiliza o MBR como aproximação. [Kang and Choy 2002] propõe alguns modelos de custo para o processamento da junção espacial. [Karam and Petry 2005] também propõe vários modelos de custo detalhados para processamento da junção espacial distribuída utilizando *R-Trees*. Estas arquiteturas armazenam as duas bases de dados R e S envolvidas na junção espacial em apenas duas máquinas, subutilizando os recursos disponíveis em um *cluster* com vários computadores.

Em [Chung et al. 2005], cada base de dados é replicada em algumas máquinas do *cluster*. Cada consulta é enviada a um servidor central e espalhada pelas máquinas que contém replicas das bases de dados envolvidas na junção espacial. [Wei et al. 2008]

replica os índices de cada base de dados entre todas as máquinas do *cluster*. Os blocos são divididos em tamanhos iguais e distribuídos uniformemente pelo *cluster*. [Xie et al. 2008] propõe um *framework* de balanceamento de carga em duas fases. A primeira fase envolve a distribuição uniforme dos dados pelas máquinas do *cluster* e a segunda fase visa balancear a carga de processamento entre os computadores. [Zhang et al. 2009] apresenta uma arquitetura que utiliza o modelo *MapReduce* para processar operações espaciais. A distribuição de dados pelas máquinas é realizada pelo *HDFS (Hadoop's Distributed FileSystem)*, que não utiliza nenhuma informação espacial para realizar esta operação. [Zhong et al. 2012] também apresenta uma arquitetura utilizando o modelo *MapReduce*. Esta arquitetura divide os dados em vários blocos, onde cada bloco contém dados próximos espacialmente, e os blocos são distribuídos no *cluster* pelo HDFS.

Todos os trabalhos apresentados anteriormente possuem técnicas de distribuição de dados para bases de dados estáticas. Estas técnicas são inviáveis para bases de dados com grandes volumes de dados e com alta frequência de atualizações. Para mitigar este problema, [Zhou et al. 2011] propôs uma técnica híbrida de distribuição de dados: a base de dados é particionada utilizando uma estratégia de distribuição estática e atualizações são tratadas com uma estratégia de distribuição dinâmica. Esta estratégia visa manter objetos próximos espacialmente em máquinas diferentes e, como dito na Seção 4.1, gera grande tráfego de dados em junções espaciais distribuídas. Em [de Oliveira et al. 2011] é proposta uma estratégia de distribuição dinâmica de dados semelhante ao algoritmo *Round-Robin*. Entretanto, esta estratégia não tenta co-localizar os dados para diminuir o tráfego de dados na rede.

Nenhuma proposta encontrada na literatura possui uma solução para processamento da junção espacial com técnicas de distribuição de dados para bases de dados dinâmicas. Os trabalhos [Zhou et al. 2011] e [de Oliveira et al. 2011] apresentam técnicas de distribuição com bases de dados dinâmicas para consultas em apenas uma base de dados.

6. Conclusões

Conforme apresentado, numa plataforma de sistemas distribuídos para processamento de grande volume de dados geográfico, a técnica de distribuição de dados é um dos principais fatores que determina a eficiência dos serviços implementados sobre uma plataforma de *middleware* para geoprocessamento distribuído. Os trabalhos da literatura apresentam propostas para bases de dados estáticas. Para sistemas com bases dinâmicas em que os dados são atualizados ou novos dados são inseridos, a técnica *Proximity Area* proposta neste artigo é uma das precursoras que apresenta bons resultados na execução de operações de Junção Espacial Distribuída num *cluster* de computadores.

Para manter o *cluster* balanceado, foi criado um fator de balanceamento, onde quanto maior o fator de balanceamento mais balanceado fica o *cluster*. Os testes demonstraram que quando o processamento é dominante, um fator de balanceamento maior (PA 0.9) permite que os recursos do *cluster* sejam aproveitados e o tempo de resposta reduzido. Quando o tráfego de dados na rede se torna dominante, um fator de balanceamento menor (PA 0.1) reduz a quantidade de mensagens trocadas na rede e, consequentemente, o tempo de resposta da junção espacial. A técnica *Proximity Area* com fator de balanceamento 0.9 apresentou o melhor desempenho entre as técnicas testadas sendo, na média, 16% melhor

que *Round Robin*.

Como trabalho futuro, será investigada uma solução híbrida, que leva em conta dois fatores de balanceamento: dados e geometrias. Esta solução híbrida visa distribuir os dados de forma com que: i) uma máquina não fique com poucos dados, pois estes servidores têm menor chance de participar da operação de junção espacial, ficando assim subutilizados; ii) uma máquina não acomode muitos objetos com geometrias complexas, pois esses servidores com muitas geometrias grandes e complexas têm maior probabilidade de serem sobrecarregados com o processamento de algoritmos de operadores topológicos. O algoritmo *Proximity Area* irá redistribuir os objetos, em determinados intervalos de tempo, para que estes sejam alocados nos servidores mais próximos espacialmente segundo os critérios de balanceamento.

Referências

An, N., Kanth, R., Kothuri, V., and Ravada, S. (2003). Improving performance with bulk-inserts in Oracle R-trees. In *VLDB-Volume 29*, page 951. VLDB Endowment.

An, N., Lu, R., Qian, L., Sivasubramaniam, A., and Keefe, T. (1999). Storing spatial data on a network of workstations. *Cluster Computing*, 2(4):259–270.

Beckmann, N., Kriegel, H., Schneider, R., and Seeger, B. (1990). The R*-tree: an efficient and robust access method for points and rectangles. *ACM SIGMOD Record*, 19(2):322–331.

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

Brinkhoff, T., Kriegel, H., Schneider, R., and Seeger, B. (1994). *Multi-step processing of spatial joins*, volume 23. ACM.

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

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.

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

de Oliveira, T., Sacramento, V., Oliveira, S., Albuquerque, P., Cardoso, M., Bloco, I., and Campus, I. (2011). DSI-Rtree - Um Índice R-Tree Escalável Distribuído. In *XXIX Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*.

Guttman, A. (1984). R-trees: A dynamic index structure for spatial searching. *ACM Sigmod Record*, 14(2):47–57.

Jacox, E. 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.

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.

Koudas, N., Faloutsos, C., and Kamel, I. (1996). Declustering spatial databases on a multi-computer architecture. *Advances in Database Technology-EDBT'96*, pages 592–614.

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.

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 GeoInformáticaâ¯ GeoInfo*, volume 2001, pages 1–8.

Schnitzer, B. and Leutenegger, S. (1999). Master-client r-trees: A new parallel r-tree architecture. In *Scientific and Statistical Database Management, 1999. Eleventh International Conference on*, pages 68–77. IEEE.

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.

Xie, Z., Ye, Z., and Wu, L. (2008). A two-phase load-balancing framework of parallel gis operations. In *Geoscience and Remote Sensing Symposium, 2008. IGARSS 2008. IEEE International*, volume 2, pages II–1286. IEEE.

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.

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.