Gain-Loss: Método de Distribuição de Dados para Processamento Distribuído de Multijunções Espaciais

Guilherme Silva Tonon[1], **Thiago Borges de Oliveira**[1]

[1]Universidade Federal de Goiás – Regional Jataí (UFG)
Rodovia BR 364 – KM 195 – 3800 – Jataí – GO – Brasil

gs.tonon90@hotmail.com, thborges@ufg.br

Abstract. *Data distribution is a challenge in the distributed execution of multiway spatial join queries. An efficient execution requires both a balanced data distribution as well as a distribution with spatial data colocalization. In this paper, we compare two methods of spatial data distribution and propose a new one called Gain-Loss, based in the R^0-tree algorithms. Our evaluation shows that Gain-Loss has a reduced area overlay between servers in all tested scenarios and also a competitive object balancing. This result indicates a more efficient execution of queries, with a reduction in the use of computational resources, mainly network usage and processing time.*

Resumo. *Um dos desafios do processamento distribuído da multijunção espacial é a distribuição dos dados de forma homogênea e colocalizada pelo cluster, de forma a obter uma execução eficiente da consulta. Neste artigo comparamos dois métodos de distribuição de dados espaciais e propomos um novo chamado* Gain-Loss*, baseado nos algoritmos da árvore R^0. Nossos experimentos mostraram que o Gain-Loss apresenta uma significativa redução da sobreposição de área entre servidores em todos os cenários, e um balanceamento de carga competitivo comparado aos demais métodos testados. Este resultado indica uma execução mais eficiente das consultas, com redução no uso de recursos computacionais, principalmente uso de rede e tempo de processamento.*

1. Introdução

O processamento de dados espaciais vem recentemente usando *clusters* de computadores para a redução do tempo de processamento das consultas espaciais, principalmente considerando grandes bases de dados [Ramirez and de Souza 2001, de Oliveira et al. 2017] e consultas de multijunção espacial. Tal consulta relaciona duas ou mais bases de dados [Mamoulis and Papadias 2001] e pode ser usada em várias aplicações cotidianas, incluindo geografia (p. ex., encontrar espécies de animais em áreas de preservação ambiental que foram atingidas por queimadas), agricultura (p. ex., avaliar a produtividade de cultivos considerando o rendimento da colheita, o uso de adubos e defensivos, a ocorrência de pragas no cultivo, dentre outros), e diagnóstico assistido por computador (p. ex., analisar imagens topológicas do cérebro para verificar se um câncer está regredindo).

Um fator de grande relevância nas consultas de multijunção espacial é a distribuição dos dados que influencia no tempo de processamento e na utilização de rede devido ao alinhamento de dados, necessário à correta avaliação do predicado da junção espacial. Por um lado, quer-se que a distribuição proporcione uma execução de consulta

balanceada, na qual todos os servidores sejam utilizados de maneira uniforme. Por outro lado, deseja-se também diminuir a necessidade de uso de comunicação entre os servidores. Estes dois aspectos foram avaliados em trabalhos recentes e evidenciou-se uma correlação inversa, ou seja, a colocalização diminui o uso de rede mas interfere negativamente no balanceamento da execução da consulta e vice-versa [Patel and DeWitt 2000, Oliveira et al. 2013, de Oliveira et al. 2017].

Duas técnicas principais são encontradas na literatura em relação à distribuição dos dados. A primeira, baseando-se em colocalização [Oliveira et al. 2013], que distribui os dados considerando a sua posição no espaço, e a segunda, baseando-se na distribuição uniforme do volume dos dados [de Oliveira 2017]. Contudo, estas duas propostas foram testadas em tipos de particionamento distintos, sendo que o particionamento tido como o mais eficiente [Patel and DeWitt 2000], chamado de disjunto, não foi avaliado juntamente com a técnica de distribuição de dados baseada em colocalização.

Neste artigo, apresentamos um estudo inicial com a avaliação do impacto da distribuição de dados no processamento de multijunções espaciais usando particionamento disjunto. Três técnicas principais de distribuição foram avaliadas: a técnica Round-Robin (RR) [de Oliveira 2017], que não considera a localização durante distribuição dos dados; a técnica Proximity-Area (PA) [Oliveira et al. 2013] que considera colocalização dos objetos; e uma nova técnica baseada no algoritmo de escolha da sub-árvore da R^0 [Xia and Zhang 2005], chamada *Gain-Loss*.

Nossos resultados preliminares, que avaliaram a distribuição resultante dos três tipos de particionamento, indicam que o método *Gain-Loss* é superior aos demais observando a redução da sobreposição de área entre os servidores, e competitivo em relação à distribuição uniforme de objetos. Resta-nos avaliar o impacto fim-a-fim, ou seja, o quanto a colocalização dos dados interfere de fato no desempenho do algoritmo de escalonamento dos fragmentos de consultas distribuídas [de Oliveira 2017] e no tempo de execução e balanceamento da execução.

2. Processamento Distribuído de Multijunções Espaciais

O uso de *cluster* de computadores possibilita o processamento de grandes bases de dados e também de consultas de multijunção envolvendo várias bases [Mamoulis and Papadias 2001]. No entanto, requer que as mesmas sejam divididas ou particionadas e posteriormente distribuídas uniformemente pelo *cluster* [de Oliveira 2017]. Dos métodos de particionamento existentes, destaca-se o método de particionamento disjunto, que proporciona uma redução da comunicação de rede durante as etapas de refinamento e filtragem da junção [Patel and DeWitt 2000]. Nele, a extensão espacial dos dados é dividida em fragmentos disjuntos, chamados de células, cada uma agrupando os objetos que estão contidos dentro de seus limites e replicando objetos que se intersectam com mais de uma célula.

Em geral, define-se a quantidade de partições de forma a permitir uma divisão uniforme da carga de trabalho pelo *cluster* quando da execução das consultas. No entanto, um número muito grande de partições pode provocar o aumento da replicação de objetos que intersectam os limites das células. Ao particionar uma base de dados, portanto, tenta-se manter um equilíbrio entre as duas situações [de Oliveira 2017].

A distribuição das partições, por sua vez, faz com que a carga de trabalho seja

similar em cada um dos servidores, evitando uma execução desbalanceada e reduzindo o tempo total de processamento da junção espacial. Se após a distribuição dos dados, objetos que se relacionam observando o predicado da consulta (interseção ou proximidade, por exemplo) estiverem em servidores diferentes, uma cópia de um dos objetos deve ser enviada para ser processada no local onde está o outro objeto relacionado. O envio é feito através da rede de comunicação do *cluster* e, em geral, devido a rede limitar a velocidade do processamento, deseja-se evitar esta comunicação organizando melhor os dados.

3. Métodos de Distribuição das Partições

3.1. *Round-Robin* (RR)

A técnica mais básica de distribuição de partições, chamada de *Round-Robin* (RR), é baseada na distribuição alternada de partições de acordo com uma lista circular de servidores do *cluster*. Este tipo de alocação faz com que os servidores recebam praticamente a mesma quantidade de partições. Apesar desta técnica provocar um aumento da comunicação no *cluster*, o efeito no balanceamento é positivo.

3.2. *Proximity Area* (PA)

O método de distribuição *Proximity Area* [Oliveira et al. 2013] (PA) busca colocalizar objetos próximos espacialmente para reduzir a comunicação na rede. O algoritmo possui um parâmetro que define o nível de balanceamento, $0 <= k < 1$, que permite alterar a intensidade com que os objetos são atraídos para um determinado servidor do *cluster*. Quando $k = 0.1$ o método PA força uma quantidade de objetos mais uniforme entre os servidores. O contrário acontece com $k = 0.9$, permitindo quantidades não-uniformes mas respeitando a colocalização.

A característica principal da técnica PA é a redução da comunicação. Quando usada juntamente com um fator k adequado, apresenta desempenho mais eficiente do que a técnica RR, quando avaliadas em particionamento não-disjunto. Experimentos mostraram que a quantidade de mensagens trocadas na rede é reduzida, o que consequentemente causa a diminuição do tempo de resposta da junção espacial [Oliveira et al. 2013]. No entanto, observou-se também que um nível muito alto de colocalização impacta de forma negativa no balanceamento da execução das consultas.

3.3. *Gain-Loss*: Distribuição de dados Baseada nos Algoritmos da R^0

A árvore R [Guttman 1984], um tipo de árvore B específico para dados multidimensionais, possui métodos heurísticos de organização dos objetos que reduzem a sobreposição dos limites geográficos dos *buckets* dos níveis superiores da estrutura (nós diretórios). Uma árvore R, assim como a B, possui um parâmetro chamado *fanout*, que define a quantidade máxima de objetos ou diretórios em cada nó. Um aprimoramento relevante da árvore R é a R^0-tree [Xia and Zhang 2005].

Nossa proposta baseia-se no algoritmo de escolha de subárvore da R^0 (aqui chamado de CHOOSE-MINOR-LOSS). Este algoritmo escolhe a sub-árvore para inserir um novo objeto observando o ganho ou perda de se inserir o mesmo em cada nó diretório da árvore, de acordo com a definição em [Xia and Zhang 2005]. Quando a sub-árvore escolhida ultrapassa o *fanout*, emprega-se um algoritmo de divisão do nó (SPLIT-RTREE-NODE), que procura uma boa divisão do nó atual, considerando também um limite mínimo de itens em cada nó para manter a estrutura de dados balanceada.

A técnica é apresentada no Algoritmo 1. Sejam O o conjunto de objetos espaciais a serem distribuídos e $servers$ um vetor com entradas que representam o conjunto de servidores do *cluster*. Primeiro, na linha 1, define-se um *fanout* de forma a forçar o algoritmo original da R^0 a criar o número necessário de servidores e define-se o número inicial de servidores em uso (linha 2). Na sequência, itera-se pelo conjunto de objetos O (linha 3) escolhendo o servidor s_i mais adequado para o objeto o, conforme o algoritmo CHOOSE-MINOR-LOSS (linha 4), adiciona-se o objeto no servidor escolhido (linha 5), aumenta-se o MBR (Mínimo Retângulo Envolvente, ou *Minimum Bounding Rectangle*) do servidor para incluir o objeto o (linha 6). Caso o número de objetos no servidor s_i seja ultrapassado (linha 7) e o número de servidores usados ainda seja menor que o total disponível, divide-se o nó chamando o método SPLIT-RTREE-NODE (linha 8) e aumenta-se a quantidade de servidores usados (linha 9).

Algoritmo 1 Algoritmo GAIN-LOSS para distribuição de objetos espaciais.

GAIN-LOSS$(O, servers)$
1 $M = \left\lceil \frac{|O|}{|servers|-1} \right\rceil$
2 $count = 1$
3 **for** $o \in O$
4 $\quad s_i =$ CHOOSE-MINOR-LOSS$(servers, o)$
5 $\quad servers[s_i].objs \mathrel{+}= o$
6 $\quad servers[s_i].mbr = servers[s_i].mbr \cup o.mbr$
7 $\quad$ **if** $(|servers[s_i].objs| > M) \&\& (count < |servers|)$
8 $\quad\quad$ SPLIT-RTREE-NODE$(servers, s_i)$
9 $\quad\quad count \mathrel{+}= 1$

4. Avaliação Comparativa dos Métodos

Para a execução dos experimentos, construímos três bases de dados sintéticas em um espaço com dimensões 100x100: i) uma base uniforme, U, com 500 retângulos distribuídos uniformemente; ii) uma base não-uniforme (*skewed*), S, com 500 retângulos distribuídos usando a lei de Zipf [Zipf 1949], com $p = 2$; e iii) uma base combinada, C, com 250 objetos gerados conforme a base S e outros 250 gerados conforme a base C. As dimensões dos retângulos gerados variaram de 1 a 10, seguindo a distribuição usada para cada base. Três tamanhos de *cluster* foram considerados: 4, 8 e 16 servidores. Os três métodos de distribuição foram considerados: *Round-Robin* (RR); *Proximity Area* (PA) com três valores de k: 0.1, 0.5 e 0.9; e o método *Gain-Loss* (GL).

Para avaliar a qualidade da distribuição dos objetos, mensuramos a sobreposição entre as regiões de cada servidor, dada pela intersecção entre a região de cada servidor para todos os pares distintos de servidores, conforme a Equação 1, sendo s a quantidade de servidores, $\cap$ representa intersecção, mbr representa o mínimo retângulo envolvente (MBR, *Minimum Bounding Rectangle*) do conjunto de objetos atribuídos ao servidor, e $area$ é uma função que calcula a área da interseção entre dois servidores i e j.

$$\eta = \sum_{i=1}^{s-1} \sum_{j=i+1}^{s} area(mbr_i \cap mbr_j) \quad (1)$$

O desvio padrão da população de quantidades de objetos em cada servidor também foi mensurado, ou seja, calculou-se o desvio padrão, σ, do conjunto $\{q(i)|1 \leq i \leq s\}$, onde q calcula a quantidade de objetos no servidor i. Um desvio padrão alto indica que pode haver uma execução desbalanceada ou, se o escalonador de consultas conseguir balancear a execução, que será necessário copiar dados de outros servidores.

A Figura 1 apresenta o resultado do experimento para sobreposição de espaço entre os servidores (η). Em (a) tem-se a sobreposição para 4, em (b) para 8 e em (c) para 16 servidores. Há um padrão nos gráficos indicando que a sobreposição decresce na seguinte ordem de métodos: RR $>$ PA 0.1 $>$ PA 0.5 $>$ PA 0.9 $>$ GL, com exceção da base C em (a) e da base U em (b), onde os métodos PA 0.5 e PA 0.9 aparecem invertidos. Naturalmente, por não considerar colocalização, o método RR apresenta a maior sobreposição. Em todos os cenários, o método GL se mostrou muito superior aos demais em relação à sobreposição, o que pode ser observado pelo pequeno tamanho da respectiva barra no gráfico.

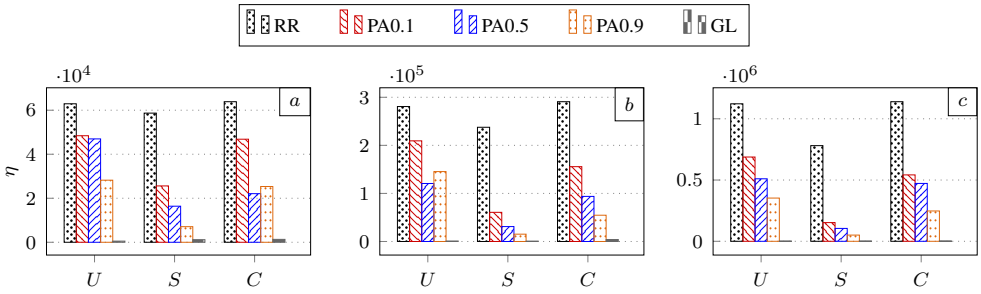

Figura 1. Sobreposição espacial entre os MBRs dos servidores. (a) **mostra a sobreposição para 4 servidores,** (b) **para 8 servidores e** (c) **para 16 servidores.**

A Figura 2 apresenta o desvio padrão, σ, da população de quantidades de objetos em cada servidor. O contrário acontece em relação à sobreposição para os métodos RR e PA, ou seja, o menor desvio ocorre para o método RR e é maior para PA 0.1, seguido de PA 0.5 e PA 0.9. O desvio padrão do método GL se mantém próximo de PA 0.5, com um pior cenário para o tamanho de *cluster* igual a 16. Como há uma boa margem para a sobreposição, apresentada anteriormente, acreditamos que é possível diminuir este valor se a quantidade mínima de itens por nó for aumentada no algoritmo original (o qual permite nós 60% não preenchidos). Testes futuros serão realizados com este cenário.

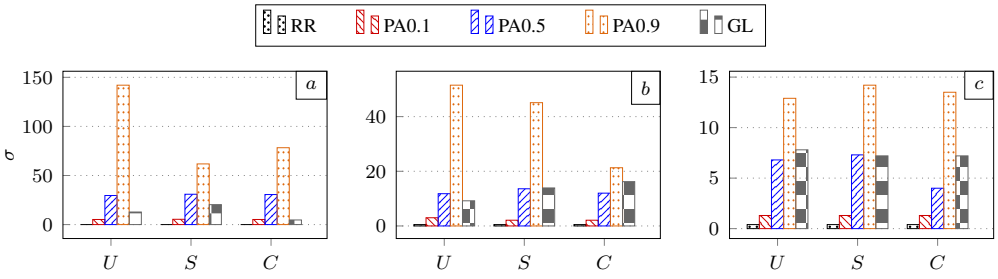

Figura 2. Desvio da população de quantidade de objetos em cada servidor conforme o método de distribuição. (a) **mostra o desvio para 4 servidores,** (b) **para 8 servidores e** (c) **para 16 servidores.**

5. Conclusão

Neste artigo propusemos um novo método de distribuição de dados espaciais chamado *Gain-Loss*, baseado nos algoritmos da árvore R^0 e o comparamos com outros dois métodos de distribuição de dados espaciais. O método proposto apresentou uma sobreposição de área entre servidores menor que os demais métodos em todos os cenários testados. Comparamos também o desvio padrão da quantidade de objetos por servidor. Nesta métrica, o novo método também se apresenta competitivo, ficando próximo do desvio padrão do método PA com $k = 0.5$. Devido a grande margem no quesito sobreposição, acreditamos que podemos melhorar ainda mais o método, aumentando o limite mínimo de preenchimento dos nós do algoritmo, o que pode resultar numa execução mais eficiente de consultas distribuídas.

Na continuação da pesquisa iremos concluir a implementação do método adaptando os algoritmos originais da R^0 para o propósito de distribuição de dados e realizaremos outros experimentos com bases de dados reais. Também analisaremos o comportamento fim-a-fim, ou seja, mensurando o impacto da distribuição de dados no tempo de execução da consulta e no uso da rede.

Referências

de Oliveira, T. B. (2017). *Efficient Processing of Multiway Spatial Join Queries in Distributed Systems*. PhD thesis, Instituto de Informática, Universidade Federal de Goiás, Goiânia, GO, Brasil.

de Oliveira, T. B., Costa, F. M., and Rodrigues, V. J. S. (2017). Distributed Execution Plans for Multiway Spatial Join Queries using Multidimensional Histograms. *Journal of Information and Data Management*, 7(3):199–214.

Guttman, A. (1984). R-trees: A Dynamic Index Structure for Spatial Searching. *SIGMOD Record*, 14(2):47–57.

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

Oliveira, S. S. T. d., Rodrigues, V. J. d. S., Cunha, A. R., Aleixo, E. L., de Oliveira, T. B., Cardoso, M. d. C., and Junior, R. R. (2013). Processamento Distribuído de Operações de Junção Espacial com Bases de Dados Dinâmicas para Análise de Informações Geográficas. In *Proc. of the XXXI SBRC*, pages 1009–1022.

Patel, J. M. and DeWitt, D. J. (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. R. and de Souza, J. M. (2001). Processamento distribuído da junção espacial. In *Anais do III Simpósio Brasileiro de Geoinformática*, pages 1–8, Rio de Janeiro, RJ.

Xia, T. and Zhang, D. (2005). Improving the R*-tree with Outlier Handling Techniques. In *Proceedings of the ACM International Symposium on Advances in Geographic Information Systems*, pages 125–134, Bremen, Germany.

Zipf, G. K. (1949). Human behavior and the principle of least effort. *Addison Wesley, Reading*.