Guilherme Silva Tonon

Métodos de Distribuição de Dados para Processamento Distribuído de Multijunções Espaciais

Jataí-Goiás

2018

Guilherme Silva Tonon

Métodos de Distribuição de Dados para Processamento Distribuído de Multijunções Espaciais

Monografia apresentada ao Curso de Bacharelado em Ciências da Computação da Universidade Federal de Goiás - Regional Jataí, como requisito parcial para obtenção do título de BACHAREL em Ciências da Computação.

Universidade Federal de Goiás - Regional Jataí - UFG-REJ

Instituto de Ciências Exatas e Tecnológicas (ICET)

Bacharelado em Ciências da Computação

Orientador: Prof. Dr. Thiago Borges de Oliveira

Jataí-Goiás

2018

Guilherme Silva Tonon
Métodos de Distribuição de Dados para Processamento Distribuído de Multijunções Espaciais/ Guilherme Silva Tonon. – Jataí-Goiás, 2018-
46 p. : il. (algumas color.) ; 30 cm.

Orientador: Prof. Dr. Thiago Borges de Oliveira

Monografia (Graduação) – Universidade Federal de Goiás - Regional Jataí - UFG-REJ
Instituto de Ciências Exatas e Tecnológicas (ICET)
Bacharelado em Ciências da Computação, 2018.

1. Distribuição de Dados Espaciais. 2. Multijunção Espacial.

Guilherme Silva Tonon

Métodos de Distribuição de Dados para Processamento Distribuído de Multijunções Espaciais

Monografia apresentada ao Curso de Bacharelado em Ciências da Computação da Universidade Federal de Goiás - Regional Jataí, como requisito parcial para obtenção do título de BACHAREL em Ciências da Computação.

Trabalho aprovado. Jataí-Goiás, data da defesa:

Prof. Dr. Thiago Borges de Oliveira
Orientador

Prof. Dra. Joslaine Cristina Jeske de Freitas
Avaliador

Prof. Me. Marcio Moraes Lopes
Avaliador

Jataí-Goiás
2018

Dedico este trabalho a minha família e a Deus, que me incentivaram e possibilitaram a realização do meu trabalho de maneira atenciosa e amorosa.

AGRADECIMENTOS

Agradeço aos meus pais, por viabilizarem a minha graduação. A minha namorada pela compreensão, solicitude e carinho. Ao meu professor orientador, pelo conhecimento científico e pessoal durante as etapas do processo de realização do trabalho.

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. Nesta monografia foram comparados os métodos de distribuição de dados espaciais Round-Robin e *Proximity Area*, além da proposta de um novo método chamado *Gain-Loss*, baseado nos algoritmos da árvore R^0. Foram feitos testes em cenário controlado, utilizando *datasets* sintéticos. A avaliação do novo método apresentou uma sobreposição de área entre servidores muito menor em todos os cenários testados e um balanceamento regular. Este resultado indica uma execução mais eficiente destas consultas, com redução no uso de recursos computacionais, principalmente uso de rede e tempo de processamento.

Palavras-chaves: *Multijunção Espacial; Distribuição de Dados; Gain-Loss.*

ABSTRACT

Data distribution is a challenge in the distributed execution of multiway spatial join queries. An efficient execution requires both a balanced data distribution in the computers of the cluster as well as a distribution with spatial data colocalization. In this monography, the spatial data distribution methods Round-Robin and Proximity Area were compared and a new one is proposed, called Gain-Loss, based in the R^0-tree algorithms. Tests were made in a controlated cenario, using sintetic datasets. The evaluation of the new method presented a very reduced area overlap between servers in all tested scenarios and also a regular object balancing. This result indicates a more efficient execution of these queries, with reduction in the use of computational resources, mainly network usage and processing time.

Key-words: *Multiway Spatial Join; Data Distribution; Gain-Loss.*

LISTA DE ILUSTRAÇÕES

Figura 1 – Representação Gráfica Matricial e Vetorial de Pontos, Linhas e Polígonos 16
Figura 2 – Polígono Representado por MBR 17
Figura 3 – *Layers* 18
Figura 4 – Particionamento Disjunto utilizando Histograma de Grade 23
Figura 5 – Particionamento Não Disjunto 23
Figura 6 – Divisão dos objetos de acordo com a quantidade de servidores 35
Figura 7 – Sobreposição espacial entre os MBRs dos servidores. (a) mostra a sobreposição para 4 servidores, (b) para 8 servidores e (c) para 16 servidores. 39
Figura 8 – Sobreposição espacial entre os MBRs dos servidores. (d) mostra a sobreposição para 32 servidores e (e) para 64 servidores. 40
Figura 9 – 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. 40
Figura 10 – Desvio da população de quantidade de objetos em cada servidor conforme o método de distribuição. (d) mostra o desvio para 32 servidores e (e) para 64 servidores. 41

LISTA DE ABREVIATURAS E SIGLAS

CPU	Central Processing Unit
DGEO	Distributed Geographic Processing System
SIG	Sistema de Informação Geográfica
MBR	Minimum Bounding Rectangle
RAM	Random Access Memory
PA 0.1	Proximity Area 0.1
PA 0.5	Proximity Area 0.5
PA 0.9	Proximity Area 0.9
RR	Round-Robin
GL	Gain-Loss

SUMÁRIO

1 Introdução 12
INTRODUÇÃO 12
2 Referencial Teórico 15
2.1 Dados Espaciais 15
2.1.1 Representação do Espaço 15
2.1.2 Objetos Espaciais 16
2.1.3 SIG 17
2.1.4 Estruturas de Indexação 18
2.1.5 Árvore R e R* 19
2.1.6 Árvore R^0 19
2.1.7 DGEO 20
2.2 Processamento Distribuído 21
2.2.1 Sistema Distribuído 21
2.2.2 Processamento Distribuído de Junções Espaciais 22
2.2.3 Particionamento dos Dados 22
2.2.4 Distribuição dos Dados 24
2.2.5 *Proximity Area* 24
2.3 Multijunção Espacial 25
2.3.1 Definição de Planos 26
2.3.2 Estimativa e Otimização de Custo 27
2.3.3 Alocação de Dados Fragmentados 28
3 Trabalhos relacionados 30
3.1 Introdução 30
3.2 Critérios de busca 30
3.3 Metodologia de análise 30
3.3.1 Proximidade 31
3.3.2 Execução em *Cluster* 31
3.3.3 Reinserção 31
3.4 Trabalhos analisados 32
3.4.1 Análise da Influência do Fator Distribuição Espacial dos Dados no Desempenho de Métodos de Acesso Multidimensionais 32
3.4.2 *Improving the R*-tree with Outlier Handling Techniques* 32
3.4.3 Processamento Distribuído de Operações de Junção Espacial com Bases de Dados Dinâmicas para Análise de Informações Geográficas 33

3.4.4 *Efficient Processing of Multiway Spatial Join Queries in Distributed Systems* . . . 33
3.5 Resumo Comparativo . . . 34
4 O Método de Distribuição *Gain-Loss* . . . 35
4.1 Introdução . . . 35
4.2 O Método . . . 35
4.3 Experimentos . . . 37
4.3.1 Métodos de Distribuição . . . 37
4.3.2 Bases de Dados . . . 37
4.3.3 *Clusters* . . . 38
4.3.4 Configuração . . . 38
4.4 Avaliação dos Resultados . . . 39
4.4.1 Sobreposição . . . 39
4.4.2 Desvio Padrão . . . 40
4.5 Considerações Finais . . . 41
5 Conclusões e Trabalhos Futuros . . . 43
5.1 Introdução . . . 43
5.2 Conclusões . . . 43
5.3 Trabalhos futuros . . . 43

Referências . . . 45

1 INTRODUÇÃO

O processamento do dado espacial emprega complexos algoritmos de geometria computacional. Recentemente, estudos tem adotado o processamento distribuído utilizando *clusters* de computadores devido ao tamanho dos *datasets*, visando a redução do tempo de processamento das consultas de multijunção espacial (BACELLAR, 2009). A distribuição dos dados é um fator de grande relevância nas operações de multijunção espacial, pois influencia no tempo de processamento e na utilização de rede. Na literatura são apresentadas duas técnicas, a primeira baseando-se em colocalização (OLIVEIRA et al., 2013), que distribui os dados a partir de sua posição no espaço, e a segunda baseando-se na distribuição uniforme do volume dos dados (OLIVEIRA, 2017). Foi desenvolvida uma nova técnica baseada no algoritmo da árvore R^0, que apresenta uma grande melhora na questão de sobreposição de objetos dos servidores. As duas técnicas propostas na literatura foram testadas em particionamento não disjunto, considerado menos eficiente (PATEL; DEWITT, 2000), e a árvore somente realiza a indexação dos objetos. Para este trabalho foram adaptadas as técnicas para a realização da distribuição de dados espaciais em sistemas distribuídos.

Tendo em vista que os dados espaciais que foram distribuídos, descritos posteriormente no corpo do trabalho, são multidimensionais, o que os torna volumosos e de tamanho extremamente variável, e que seus relacionamentos são complexos, eles se mostram propícios para o processamento distribuído (CIFERRI, 2002). Quando se faz uso de muitos conjuntos de dados os algoritmos de junção espacial começam a ter um processamento intenso, além de poderem utilizar muito da rede que está disponível para o *cluster*.

Portanto, distribuição é um fator determinante para o paralelismo quando utiliza-se de *clusters* (OLIVEIRA et al., 2013), sendo que a mesma deve ser feita de forma balanceada, para que todos os servidores sejam utilizados de maneira uniforme. A colocalização também tem grande influência, pois um servidor necessita ter todos os dados necessários para poder processar uma operação, caso contrário necessitará da utilização da banda de rede disponível para troca de informação. Então, embora a colocalização diminua a necessidade do uso da rede, ela também pode interferir de forma negativa, diminuindo o paralelismo no processamento.

A multijunção espacial, por sua vez, é uma das consultas espaciais mais importantes, pois relaciona dois ou mais *datasets* em relação a um predicado espacial específico (OLIVEIRA; COSTA; RODRIGUES, 2015), tornando-a de certa forma complexa. Sendo assim, para a realização de multijunção distribuída é comum o uso de *clusters* de computadores, com a intenção de aumentar o poder computacional sem crescer demasiadamente os custos

financeiros (BACELLAR, 2009). Uma consulta de multijunção espacial pode ser realizada, por exemplo, utilizando dois *datasets* distintos - rios e estradas. A junção espacial é realizada a partir de um predicado, como por exemplo interseções, o que resultaria em locais onde possivelmente devem existir pontes.

Sendo a distribuição dos dados de grande importância no processamento das junções, pois tem grande relevância em relação ao consumo de recursos computacionais que elas demandam, levantam-se algumas questões que são respondidas com este trabalho:

1. Qual o impacto dos diferentes níveis de balanceamento no processamento da multijunção espacial?
2. Colocalizar os dados é um método de distribuição mais eficiente do que simplesmente balancear os dados de forma aleatória?
3. É possível determinar um ponto de equilíbrio entre colocalização e balanceamento?
4. Existe um nível de colocalização a ser considerado mais adequado a todas consultas distintas?

Para a realização dos experimentos comparativos foram identificados métodos de distribuição e implementados para o cenário onde foram realizados os experimentos. Foram analisados diferentes níveis de colocalização na distribuição dos objetos, e adaptadas algumas técnicas propostas na literatura para aprimorar a colocalização da distribuição. Para análise dos resultados obtidos estão disponíveis gráficos, divididos entre as métricas de sobreposição e desvio padrão.

Como contribuições, este trabalho apresenta:

1. A proposta de um método de distribuição de dados para multijunções espaciais baseado nos algoritmos da R^0; e
2. Uma avaliação comparativa de três métodos de distribuição utilizando *datasets* espaciais sintéticos.

A partir da análise dos resultado foi possível observar uma peculiaridade, na maioria dos casos, sobre os níveis de balanceamento no método *Proximity Area* em relação a sobreposição. Colocalizar os dados de forma eficiente trouxe resultados muito bons no novo método proposto, porém em alguns casos isso casou um impacto significativo no balanceamento. Acredita-se que ainda não foi possível identificar o ponto ideal de equilíbrio entre paralelismo e colocalização, porém como os resultados em relação a sobreposição são tão promissores, entende-se que é possível fazer alterações que tornarão o método proposto mais hábil no processamento das operações de forma paralela.

O trabalho está organizado da seguinte maneira: No Capítulo 2, é apresentado o referencial o referencial teórico, onde são detalhados os conceitos utilizados, bem como os trabalhos em que se basearam as técnicas de distribuição empregadas no novo método proposto e o funcionamento dos métodos em si. No Capítulo 3, são apresentados os trabalhos relacionados e os critérios utilizados para busca e análise, contando também com um resumo comparativo entre eles. No Capítulo 4, são detalhados como ocorreram os experimentos, os dados e métodos utilizados, os resultados e como eles foram avaliados. Por ultimo, no Capítulo 5 estão as conclusões obtidas através da avaliação dos resultados e os possíveis trabalhos futuros.

2 REFERENCIAL TEÓRICO

Para melhor compreensão da monografia, neste capítulo serão abordados temas, métodos e conceitos fundamentais para o processamento de operações espaciais, que foram utilizadas nos experimentos para validação da eficácia dos métodos. A Seção 2.1 aborda os dados espaciais e explica como ocorre a representação do espaço e objetos presentes, além de explicar o funcionamento dos SIGs (Sistemas de Informação Geografica), explicar sobre as estruturas de indexação utilizadas e apresentar a suíte DGEO. A Seção 2.2 apresenta conceitos sobre processamento distribuído de dados, e como é possível aplicar a distribuição do processamento em operações de multijunção espacial, apresentando o método *Proximity Area* e o método *Gain-Loss* da árvore R^0. A Seção 2.3 explica a multijunção espacial, detalhando todo o processo de definição de planos de modo a otimizar o processamento da consulta.

2.1 Dados Espaciais

Dado espacial é um tipo de dado específico, manipulado com técnicas especializadas por banco de dados espaciais. O termo é sinônimo de dados geográficos ou geoespaciais, se referindo a objetos perto ou na superfície da terra (OLIVEIRA, 2017). São utilizadas estruturas conceituais pré-concebidas para criar a representação de um espaço geográfico passível de análise e tratamento (FITZ, 2018).

Dados espaciais podem representar qualquer referência do espaço que queira ser estudada e analisada, por exemplo, um quarto específico, toda uma casa, o nosso planeta, outros planetas do universo ou até mesmo uma pequena placa de circuitos elétricos. Em qualquer uma destas situações, os dados espaciais presentes na área de interesse são convertidos, para que possa ser feita representação planar do espaço.

2.1.1 Representação do Espaço

Dados espaciais de modo geral são divididos em objetos e podem ser armazenados em estruturas matriciais, como exemplificado do lado direito da Figura 1, onde o espaço é dividido em células de mesmo tamanho, designadas em linhas e colunas. Cada célula armazena os dados das propriedades espaciais dos objetos contidos em si (OLIVEIRA, 2017).

Outra forma de representação, por modelos vetoriais, é demonstrada no lado esquerdo da Figura 1, onde a posição associada a cada pixel na imagem corresponde a coordenadas específicas (x,y) que são armazenadas. As entidades espaciais são apresentadas

na forma de pontos, linhas e polígonos, que representam a forma dos objetos e os fenômenos associados a superfície. (FITZ, 2018).

Os pontos contém somente um par de coordenadas associadas a ele, e são comumente utilizados para representar a localização de algo no espaço. As linhas são apresentadas utilizando uma série de pontos conectados entre si e são comumente utilizadas para representar rios, estradas, fronteiras, e outros. Os polígonos são apresentados pela união de duas ou mais linhas, visto que o ponto de começo da primeira linha coincide com o ponto final da última linha e são comumente utilizados para representar objetos com área, como edificações, cidades, lagos e mares, por exemplo.

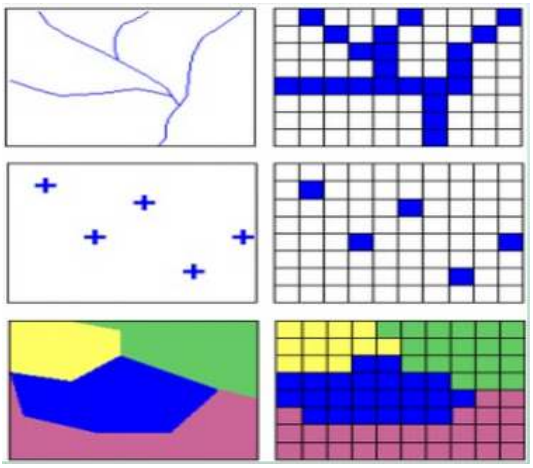

Figura 1 – Representação Gráfica Matricial e Vetorial de Pontos, Linhas e Polígonos
Fonte: Adaptado de https://bit.ly/2m6dCvg

Embora ambos os modelos de representação tenham vantagens e desvantagens na sua utilização, dependendo do propósito da aplicação é possível definir o mais adequado. Neste trabalho a abordagem utilizada será a vetorial, pois conta com maior nível de precisão na localização do objeto e topologia inerente, simplificando a análise espacial (OLIVEIRA, 2017).

2.1.2 Objetos Espaciais

Um objeto espacial é a representação de uma entidade no espaço, tendo atributos que o descrevem, tanto por sua localização e tamanho, quanto pelas relações espaciais como adjacência ou sobreposição (OLIVEIRA, 2017).

Para a realização de operações espaciais, os objetos contidos no espaço a ser analisado são simplificados em retângulos, utilizando a técnica do MBR (*Minimum Bounding Rectangle*), que é um modo de representação que se aproxima da representação do objeto original. A técnica define as medidas dos retângulos a partir dos pontos mais extremos da topologia original do objeto. Na Figura 2, objeto original esta representado por um

polígono irregular, e os seus pontos mais extremos determinam as medidas do MBR formado ao seu redor.

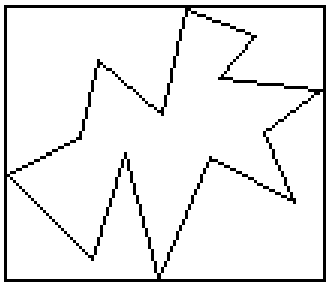

Figura 2 – Polígono Representado por MBR
Fonte: Adaptado de https://bit.ly/2MTaPl6

No processamento da multijunção espacial, o MBR é utilizado para limitar a quantidade de dados sobre um determinado objeto na etapa de filtração, e posteriormente é desconsiderado para a realização da etapa de refinamento.

2.1.3 SIG

Um SIG é um sistema alimentado por dados espaciais com função de análise geoespacial. Ele pode assumir funções diversas, por isso existem SIGs mais generalizados e outros para fins mais específicos. Embora cada sistema possa exercer funções distintas, é possível destacar algumas funções comuns aos sistemas em geral:

- aquisição e edição de dados;
- gerenciamento do banco de dados;
- análise geográfica de dados;
- representação de dados.

A base de dados utilizada deve ser representada de forma compreensiva, utilizando objetos espaciais, métodos de acesso para recuperação rápida, linguagens de consulta e algoritmos relacionados ao processo de representação e análise de dados (OLIVEIRA, 2017).

Para trabalhos científicos e análises espaciais, possivelmente a função mais importante de um SIG é a análise geográfica dos dados (FITZ, 2018). A análise espacial consiste na manipulação dos dados, de forma que as informações decorrentes dessa manipulação sejam utilizadas para o auxílio da tomada de decisões e até mesmo revelar informações que antes não eram aparentes (OLIVEIRA, 2017).

Um SIG bem programado pode realizar diversas análises, entretanto deve estar alimentado de dados espaciais, que serão divididos em camadas em sobreposição denominadas *layers*. Cada *layer* é uma representação geográfica do espaço.

Um *layer* pode conter um grande número de objetos espaciais do mesmo tipo, desde que tenham a mesma estrutura e sejam representados na mesma localização geográfica específica. Cada um dos objetos é descrito pelas suas características e seus atributos espaciais (OLIVEIRA, 2017).

Na Figura 3 são exemplificados *layers*, que estão divididos tendo a superfície terrestre como referência. Em cada uma das camadas estão contidos objetos relacionados. A sobreposição e análise destas camadas pode gerar informações adicionais as pré-existentes (FITZ, 2018).

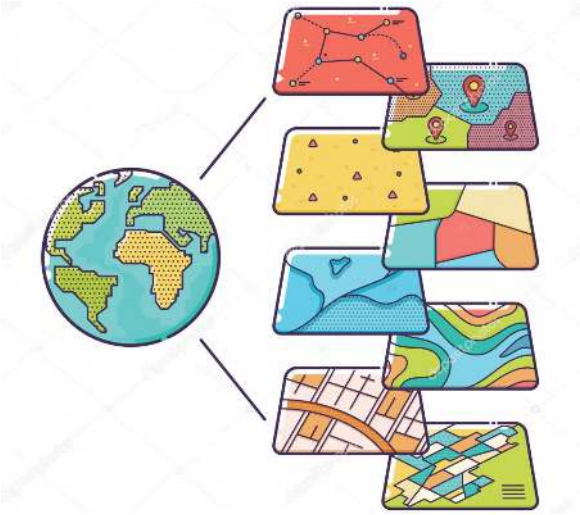

Figura 3 – *Layers*
Fonte: Adaptado de https://bit.ly/2MRsMR0

2.1.4 Estruturas de Indexação

Para a utilização dos SIGs e suas funções de analise espacial é preciso relacionar os dados ao mesmo, para que seja feita a recuperação das informações sobre os objetos. Conforme citado anteriormente na Seção 2.1, os dados espaciais são complexos de se manipular pelas suas características implícitas e também seu volume.

Para que o desempenho na recuperação destes objetos seja melhor, são implementadas estruturas de dados e algoritmos de pesquisa, os métodos de acesso multidimensionais (MAM). O intuito é que para a recuperação de objetos contidos numa área especifica, não seja necessário a analise do conjunto completo de objetos (CIFERRI, 2002).

Exite um grande número de MAMs com diferentes características propostos na literatura. Seus desempenhos variam de acordo com o tipo de dados utilizado e o tipo de consulta. Em seu trabalho, Ciferri (2002) realiza um levantamento dos principais MAMs

e faz experimentos no caráter de identificar qual deles é mais eficiente em determinados tipos de consulta, analisados através de uma técnica experimental de *benchmark*. A partir da análise do desempenho dos diferentes MAMs, é possível observar grande proeminência dos membros da família da árvore R.

2.1.5 Árvore R e R*

A árvore R (GUTTMAN, 1984), um tipo de árvore B específica 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.

Pela sua ampla popularidade, a árvore R foi amplamente pesquisada e foram propostas diversas adaptações para melhorar seu desempenho, principalmente para conjuntos de dados maiores e consultas mais complexas. Foram propostos novos métodos de particionamento dos nós e tratamento de objetos fora do padrão, além da adaptação para estruturas paralelas (CIFERRI, 2002).

A árvore R* (BECKMANN et al., 1990) foi proposta como modificação da árvore R para proporcionar melhor desempenho na manipulação dos objetos. A sua estrutura se asemelha a sua antecessora, com algumas diferenças como novos critérios para recuperação de objetos (*coverage, overlap, margin e storage*) e mudanças no algoritmo de inserção.

Posteriormente em sua pesquisa, Xia e Zhang (2005) demonstram algumas alterações em métodos já existentes na árvore R*, e o desenvolvimento de novas métricas para avaliar a qualidade dos objetos. Esta nova variação da árvore R* é chamada de árvore R^0, e é descrita na Subseção 2.1.6

2.1.6 Árvore R^0

A partir da árvore R*, Xia e Zhang (2005) propuseram uma nova estrutura de indexação que buscava ser mais eficiente que sua antecessora em operações espaciais. Esta estrutura, chamada de árvore R^0, tem como principal característica a boa gerencia de *outliers*, que são objetos considerados muito grandes ou localizados distantes dos demais.

A árvore R^0 possui reinserção distinta da árvore R*, ela força a reinserção de objetos previamente inseridos quando um nó atinge seu número máximo de objetos. A quantidade de objetos que vai ser realocada não é fixa, porém é controlada a partir da quantidade total de objetos. Combinado com novos métodos de identificação de objetos, a árvore R^0 consegue avaliar quais os melhores objetos para trocar de lugar.

Para a otimização das operações espaciais, os MBRs envolvidos devem ter a menor área possível e serem quadráticos, pois um MBR muito grande ou longo e fino tende a

intersectar com outros MBRs. Para a identificação dos objetos considerados *outliers* a estrutura conta com novos métodos, chamados de qualidade, ganho e perda.

O calculo da qualidade é associado a um MBR, que pode conter diversos objetos. A fórmula utilizada para o calculo é descrita na Equação 2.1, que retorna a qualidade de um MBR r, dados sua largura l e altura a, e uma constante α, com valor entre 0 e 1.

$$Q(r) = \frac{1}{l * a} * \left(\frac{min[l, a]}{max[l, a]} \right)^{\alpha} \tag{2.1}$$

Avaliada a qualidade dos MBRs, quando a árvore realizar a expansão de algum nó, será realizada a reinserção de objetos. Para isto ela utiliza a fórmula de ganho e perda. Dado um MBR m_1, o ganho de se retirar um MBR m_2 que esteja espacialmente contido em m_1 é definido pela Equação 2.2. A perda de expandir um MBR m_2 para m_1 tem o mesmo valor do ganho de se retirar um MBR m_1 de m_2.

$$G(m_1, m_2) = 1 - \frac{Q(r_1)}{Q(r_2)} \tag{2.2}$$

2.1.7 DGEO

As análises espaciais foram feitas por meio de uma aplicação descrita no trabalho de Oliveira (2017) chamada DGEO. Esta suíte foi implementada utilizando as linguagens de programação C e Go. A aplicação utiliza técnicas de paralelismo e distribuição de dados, possui métodos e algoritmos originais e também utiliza a biblioteca GEOS (*Geometry Engine - Open Source*). A aplicação processa consulta de junção espacial sobre *datasets* espaciais.

Além de ser capaz de gerenciar de modo eficiente as consultas espaciais distribuídas, a suíte DGEO também é capaz de simular o processo de junção espacial, utilizando dados reais inseridos no sistema, a configuração da consulta desejada e as características do *cluster*. São utilizadas equações para realizar a estimativa dos custos relacionados a consulta, tendo como retorno os valores de custo de processamento da CPU e a quantidade de dados transmitidos pela rede.

Para a realização deste trabalho o DGEO foi modificado, sendo adicionado os novos métodos de distribuição de dados *Proximity Area* e *Gain-Loss*, para que possa ser explorada uma nova maneira de distribuir os dados pois o DGEO utiliza o algoritmo Round-Robin para distribuição dos dados, que não leva em consideração a localização dos objetos no espaço.

No trabalho de Oliveira et al. (2013) foi demonstrado que a colocalização dos objetos aplicada de forma especifica pode melhorar o desempenho de junções espaciais comparativamente ao método Round-Robin.

Também foi demonstrado por Ciferri (2002) e Xia e Zhang (2005) a eficiência das árvores R* e R^0 na indexação de objetos espaciais. Então, para a execução em um ambiente *clusterizado*, os métodos utilizados da R^0 tiveram de sofrer adaptações quanto ao particionamento dos nós e quantidade máxima de objetos para particionar, que serão mais detalhadamente descritas na Subseção **??**.

2.2 Processamento Distribuído

Como os *datasets* comumente tem uma grande quantidade de dados e na multijunção espacial vários *datasets* são utilizados em uma mesma consulta, a utilização de somente um computador para o processamento se torna uma alternativa ruim, pois exige maior capacidade de processamento. Uma solução viável é a utilização de *clusters* de computadores para que a consulta seja realizada de forma distribuída em vários computadores, em um tempo reduzido (BACELLAR, 2009).

2.2.1 Sistema Distribuído

Um sistema distribuído, ou *cluster*, pode ser definido como um conjunto de computadores independentes entre si, porém apresentando-se como um único sistema. Para que esse conjunto de computadores se comporte como um sistema único, é utilizado um *middleware* controlado por um computador mestre que realiza a comunicação entre os múltiplos hardwares (TANENBAUM; STEEN, 2007).

Computação em *cluster* foi popularizada quando o preço de computadores pessoais e *workstations* se tornaram muito altos para atingir a performance exigida. Pode ser mais viável construir um super computador utilizando um conjunto de computadores mais simples conectados a uma rede de alta velocidade.

Para que um sistema distribuído se apresente como um sistema único é preciso de que seja transparente. Existem diferentes formas de transparência, todas implicando na capacidade de esconder que os recursos se encontram distribuídos em diferentes lugares. De acordo com Tanenbaum e Steen (2007), a transparência pode ser aplicada em diferentes aspectos:

- acesso a representação dos dados
- localização de recursos e dados
- migração de recursos e dados
- realocação de recurso durante o uso
- replicação de recursos e dados

- falhas

2.2.2 Processamento Distribuído de Junções Espaciais

O processamento distribuído de junções espaciais é fundamental devido ao tamanho dos *datasets* presentes em uma consulta. Para realizar o processamento das consultas de forma distribuída, é necessário que os objetos que são relacionados entre si estejam presentes no mesmo computador, portanto os objetos presentes nos diferentes *datasets* devem ser particionados e distribuídos uniformemente pelo *cluster* (OLIVEIRA, 2017), de modo que a quantidade de carga seja similar em cada um dos computadores. Caso algum computador do *cluster* tenha que realizar uma quantidade muito maior de processamento do que outros, o tempo de resposta da junção espacial aumentará.

Se após a distribuição dos dados, objetos que se relacionam estiverem em locais diferentes, uma cópia de um dos objetos deve ser enviada para ser processada no local onde está o outro objeto com que ele se relaciona. O envio é feito através da rede disponível para o *cluster*.

2.2.3 Particionamento dos Dados

É possível dividir a representação do espaço pelos objetos nela contidos ou por conjuntos de células ou *buckets* que armazenam dados escalares. No caso da divisão por espaço, cada uma das células armazena dados sobre uma parte especifica do *dataset* referência (ROH et al., 2010).

A maioria dos algoritmos que realizam esse processamento utilizam destes dois métodos distintos para o particionamento dos dados:

- Particionamento Disjunto: Divide o espaço em fragmentos disjuntos, chamados de células, onde cada célula agrupa os objetos que intersectam com seus limites, replicando objetos que intersectam com mais de uma célula. Na Figura 4 é possível observar um exemplo de particionamento disjunto. O*dataset* é dividido em células, sendo que a intensidade da cor determina a quantidade de objetos que intersectam com aquela célula.

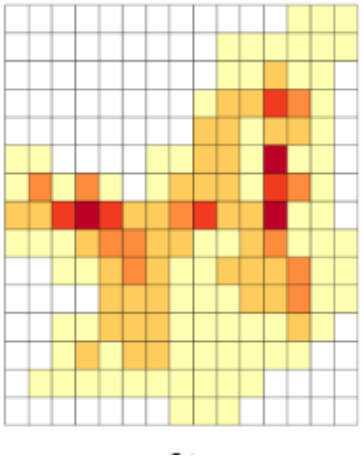

Figura 4 – Particionamento Disjunto utilizando Histograma de Grade
Fonte: (OLIVEIRA, 2017)

- Particionamento Não Disjunto: As partições, neste caso chamadas de *buckets*, podem se sobrepor de modo a acomodar um ou mais objetos em sua extensão total, não necessitando da replicação de objetos. Na Figura 5 é exemplificado o particionamento não disjunto através dois *datasets* que se sobrepõem. Embora a área coberta por dois servidores ($S1$ e $S4$) não tenham sobreposição entre si, a área dos servidores $S2$ e $S3$ se intersectam mesmo não tendo objetos em comum, devido a posição dos objetos contidos neles.

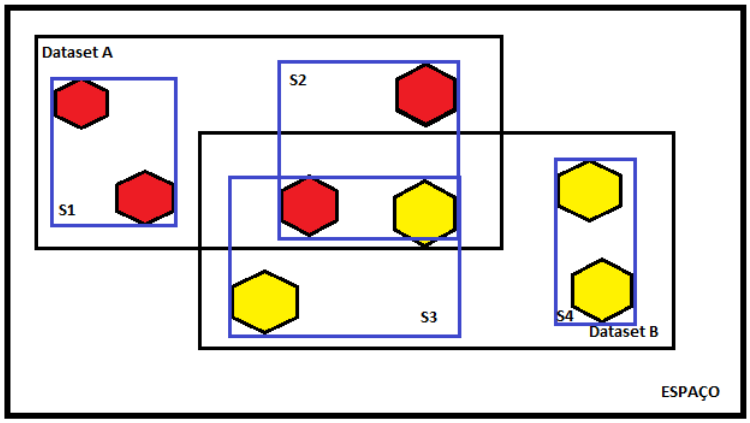

Figura 5 – Particionamento Não Disjunto
Fonte: Adaptado de (OLIVEIRA et al., 2013)

Este trabalho utiliza o método de particionamento disjunto, pois como é possível constatar na literatura (PATEL; DEWITT, 2000), o método de particionamento não disjunto provoca aumento na comunicação de rede durante o refinamento e filtração da junção, o que pode levar ao aumento do tempo de processamento total.

2.2.4 Distribuição dos Dados

Para realizar a distribuição das partições dos dados, podem ser utilizados métodos diferentes. O método mais comum é o Round-Robin, que conquiste em distribuir as partições de forma uniforme, balanceando a quantidade de dados pelo *cluster* que realizara o processamento da junção. Como o método Round-Robin desconsidera a colocalização das partições, comumente objetos que se sobrepõem estão em computadores diferentes no *cluster*. É preciso que os dois objetos do par estejam no mesmo computador, então um objeto é replicado e transferido por meio da rede (OLIVEIRA, 2017).

2.2.5 *Proximity Area*

Outro método de distribuição descrito no trabalho de Oliveira et al. (2013) é chamado *Proximity Area* que busca colocalizar objetos próximos espacialmente. O algoritmo possui um nível de balanceamento, podendo alterar a colocalização dos objetos em um fator de 0,1 a 0,9. Agrupando os objetos no *cluster* baseado na sua posição no espaço faz com que a chance de objetos que se relacionam pelo predicado da junção estarem no mesmo computador aumentar. Caso os objetos estejam no local adequado para seu processamento, não há necessidade de troca de informações pela rede.

Mesmo que o balanceamento de carga entre os computadores do *cluster*, que é uma característica do método Round-Robin, seja um fator importante para o processamento da junção pois aumenta o grau de paralelismo, a técnica *Proximity Area* utilizada em certo valor de colocalização apresenta desempenho mais eficiente do que Round-Robin, pois diminui a quantidade de mensagens trocadas na rede, levando a diminuição do tempo de resposta da junção espacial (OLIVEIRA et al., 2013). Um dos objetivos deste trabalho é analisar qual o impacto da colocalização no tempo de processamento de multijunções espaciais, pois é possível observar que um nível muito alto de colocalização pode impactar de forma negativa nas consultas espaciais.

A técnica *Proximity Area* está descrita no Item 2.2.5 e seu objetivo é escolher o servidor S no qual o novo objeto O será alocado, tendo três parâmetros para serem levados em consideração:

1. o MBR de O,
2. o fator de balanceamento a ser considerado,
3. uma lista obtida através do servidor monitor contendo informações sobre os servidores do *cluster*

O monitor armazena, de cada máquina, o particionamento da grade dos histogramas e a quantidade de objetos contidos na grade. Cada elemento I da lista possui como atributos:

1. o particionamento determinado que contém todos os objetos de S,
2. uma referência para o servidor S correspondente a I,
3. o número de objetos em S

PROXIMITY AREA($M, k, lista$)

```
1  min ← inteiro máximo
2  for cada elemento I em lista
3       if I.numObjetos = 0
4            return I.referenciaServidor
5       if I.numObjetos < min
6            min ← I.numObjetos
7  minArea ← número de ponto flutuante máximo
8  referenciaServidor ← null
9  for cada elemento I em lista
10      if (min/I.numObjetos) > k
11           area ← aumento de área de I.MBR para inserir M
12           if area < minArea
13                minArea ← area
14                referenciaServidor ← I.referenciServidor
15 return referenciaServidor
```

2.3 Multijunção Espacial

Para que sejam obtidas informações adicionais às pré-existentes nas camadas de dados, é preciso que sejam realizadas técnicas de análise de dados espaciais, chamadas de consultas espaciais. Um tipo de consulta espacial é a junção espacial complexa, ou multijunção espacial, que utiliza dois ou mais *layers* em suas análises. Um exemplo deste tipo de consulta espacial é: "todas as espécies de animais ameaçados de extinção, que vivam na área da floresta amazônica e habitem rios com distância maior do que 3 metros de margem a margem". Devem ser considerados pelo menos três *datasets* nesta consulta: animais ameaçados de extinção, floresta amazônica e rios. Serão avaliados os objetos contidos na consulta e os que forem compatíveis as exigências serão retornados como resultado.

A multijunção espacial realiza uma consulta em razão de um predicado e várias camadas de dados, podendo ser definida formalmente da seguinte maneira: dado um conjunto de n bases de dados *R1, R2, ..., Rn* e uma consulta Q, onde *Q(i,j)* é o predicado espacial que deve ser aplicado entre *Ri* e *Rj*, retornar todas as *n-uplas ((r(1,2), ..., r(n,z))* que satisfaçam a condição do predicado espacial *Q(i,j)* (MAMOULIS; PAPADIAS, 2001).

Dada esta definição, podemos representar a consulta Q como um grafo $G = (N, E)$ onde os vértices N corresponderiam às variáveis do problema (base de dados), e as arestas E corresponderiam às restrições espaciais binárias (predicados de junção espacial) (CUNHA et al., 2014). Alguns dos predicados espaciais são:

- Intersecção: retorna características comuns dentre os *datasets* alvo relacionados a um local do *dataset* fonte.
- Contido em uma Distância de: retorna todas as características comuns em outros *datasets* dentro de uma distância especificada no *dataset* fonte.
- Contém: retorna características do *dataset* alvo contidas no *dataset* fonte.
- Contém Completamente: retorna características do *dataset* alvo contidas totalmente no *dataset* fonte, ignorando características coincidentes ou com limites excedidos.
- Idêntico: retorna características de mesma posição geográfica.
- Atravessado por Contorno: retorna características que compartilham somente um vértice.
- Compartilha Segmento de Linha: retorna características que compartilhem dois ou mais vértices.

Para a formação dos pares que serão utilizados na análise, são divididas duas etapas distintas no processo de junção espacial. A primeira etapa consiste na **filtração** dos pares, onde são selecionados os possíveis candidatos a resposta. Em função da utilização do MBR para representação dos objetos, é preciso que seja aplicada a segunda etapa do processo, pois o MBR fornece uma representação não exata do objeto. Já a segunda etapa é baseada no **refinamento**, onde o algoritmo verifica os pares candidatos para identificar se eles realmente satisfazem o predicado da junção que está sendo realizada, porém agora utilizando as dimensões reais dos objetos (OLIVEIRA, 2017).

Uma consulta espacial, mesmo preservando a semântica da consulta, pode ter vários planos de execução diferentes que utilizam de algoritmos distintos para realizar o processamento da junção. Para a escolha do plano mais adequado e de menor custo para a realização da operação espacial, foram utilizadas técnicas descritas na Subseção 2.3.1.

2.3.1 Definição de Planos

Existem diferentes formas de relacionar *datasets* durante consultas. É possível combinar *datasets* em cadeia, relacionado-os em pares sem repetição, também é possível relacionar de modo a formar um ciclo entre os *datasets* ou então relacionar todos os *datasets*

entre si, que é o tipo mais complexo de relacionamento, chamado de *clique* (OLIVEIRA, 2017).

Mamoulis e Papadias (2001) demonstraram que em junções espaciais processadas em série, dependendo do tipo da consulta, número de *datasets* envolvidos e número de algoritmos de junção envolvidos, podem haver múltiplos planos de execução. Por exemplo, em uma junção com cinco *datasets* relacionados em *clique* e com três algoritmos de junção diferentes, existem quase 100 combinações de planos de execução. Todos os planos preservam a mesma semântica, contudo podem levar tempos diferentes para chegarem no resultado final.

Para facilitar ou melhorar a escolha do plano de execução mais adequado, é possível utilizar um otimizador de consultas, que estima o custo computacional através de equações. (OLIVEIRA; COSTA; RODRIGUES, 2015).

2.3.2 Estimativa e Otimização de Custo

Um otimizador considera os aspectos dos *datasets* envolvidos para definir um plano de execução que determine a ordem de processamento e como serão relacionados os *datasets*, e os algoritmos que serão utilizados em cada passo. A procura não é pelo melhor plano de execução, pois dependendo da quantidade de variáveis envolvidas na consulta isto pode se tornar uma tarefa muito difícil, mas sim por um plano com tempo de execução razoável (OLIVEIRA, 2017).

Foram definidas por Mamoulis e Papadias (2001) algumas equações para estimar o custo computacional de junções espaciais complexas antes do seu processamento. Para que estas equações sejam precisas em dados reais, é necessário aplicá-las sobre um histograma multidimensional de grade exemplificado na Figura 4. Só é possível se aplicar a estimativa de custo devido ao fatos dos histogramas serem uma ferramenta de representação aproximada dos dados multidimensionais em um espaço (ROH et al., 2010).

Além dos histogramas de grade fornecerem uma forma comum de particionar os dados espaciais, suas principais vantagens são que a sua construção é simplificada, a sua eficiência no tempo de estimar consultas e manutenção incremental para *datasets* não estáticos.

Embora exista uma taxa de erro grande para dados não uniformemente distribuídos no espaço do *dataset*, é possível diminuir esta taxa aumentando o número de células que dividem o histograma. Em contrapartida, aumentando o número de células e consequentemente o detalhamento, aumenta-se também a quantidade de memória necessária para armazenar o histograma, bem como o fato de que um objeto intersectar diferentes células pode aumentar a quantidade de erros em relação aos seus limites (OLIVEIRA, 2017).

No fim das contas, ainda é mais vantajoso a utilização de um número maior de

células, pois como a consulta é realizada de forma distribuída, uma maior quantidade de células gera um maior particionamento e consequentemente um maior balanceamento dos dados no *cluster*, reduzindo o efeito causado pelos dados não uniformemente distribuídos, além de aumentar o grau de paralelismo no processamento da consulta.

Existem algoritmos que otimizam a consulta baseados em uma regra definida. Mesmo selecionando um plano de maneira rápida, otimizadores baseados em regra são menos flexíveis a adição de novos algoritmos de junção e têm limitações na análise de um *dataset* cujas propriedades não são consideradas em seu *design*. Contudo, existem algoritmos que otimizam a consulta baseado no custo de cada plano. Estes são mais flexíveis e adaptativos do que os baseados em regra. É estimado o custo para cada um dos diferentes planos, levando em consideração tanto propriedades do *dataset*, quanto custos de entrada e saída de dados, bem como custo de uso da CPU (Central Processing Unit). Por terem que calcular o custo de todos os planos possíveis, algumas consultas com mais opções de planos podem ter resposta mais demorada. (OLIVEIRA, 2017).

No trabalho de Oliveira, Costa e Rodrigues (2015) é proposto um algoritmo que estima o custo de execução do processamento de junções espaciais utilizando histogramas multidimensionais e também um método de criação de histogramas que se aliados apresentam uma assertividade de melhora nos resultados das consultas com *datasets* reais com que foram testadas.

2.3.3 Alocação de Dados Fragmentados

Levando em consideração que os dados inicialmente estão distribuídos pelo *cluster*, é função do otimizador decidir onde e quando serão executados determinados fragmentos da consulta e consequentemente de e para onde os dados devem ser copiados.

Para o auxílio da tomada desta decisão, o otimizador se baseia em alguns fatores:

- Designar um processamento menor aos servidores que tem partições maiores alocadas;
- Designar o processamento de um modo que a conexão de internet não limite os processadores a ponto deles ficarem ociosos;
- Designar o processamento de modo à manter o balanceamento dos dados no *cluster*.

É possível gerar um plano para esta movimentação de dados que pode ser feito em diferentes estágios durante o processo de otimização. A literatura propõe diferentes métodos para a realização destes planos. É possível realizar o cálculo de todos os planos de movimentação de dados possíveis, dados os planos de execução originais da junção. Outra abordagem é especificar métodos de processamento para servirem de base a partir das informações iniciais e gerar planos para cópia e processamento dos dados em tempo

de execução. Alguns métodos foram desenvolvidos com foco em processos que utilizam *clusters* de tamanho moderado e partições de dados não muito volumosas e buscando encontrar a melhor solução somente localmente, desconsiderando o contexto global do processamento (ÖZSU; VALDURIEZ, 2011).

No trabalho de Oliveira (2017) são propostos algoritmos baseados na teoria da otimização combinatória, que foram especificamente desenvolvidos para o contexto de um alto número de máquinas no *cluster* e múltiplos *datasets* com grande volume de dados, demonstrando ser mais adequado no contexto das multijunções espaciais.

3 TRABALHOS RELACIONADOS

3.1 Introdução

Serão apresentados neste capítulo alguns dos trabalhos relacionados mais relevantes para o desenvolvimento desta monografia e do novo método descrito. Os artigos foram retirados de bases de dados educacionais como os repositórios da Universidade Federal de Goiás, Universidade Federal de Pernambuco, Universidade Estadual da Pensilvânia e da Universidade de Brasília.

De modo a refinar a busca por trabalhos, foi realizada uma revisão sistemática descrita detalhadamente na Seção 3.2. A revisão resultou uma grande quantidade de trabalhos que foram posteriormente ranqueados a partir de sua aplicabilidade nesta pesquisa.

Alguns são considerados critérios fundamentais da implementação para analise dos trabalhos relacionados, que são métodos de otimização dos resultados. Estes critérios estão presentes separadamente nos trabalhos apresentados e reunidos nesta monografia.

Um resumo da analise dos critérios em virtude dos trabalhos relacionados, é apresentado na Tabela 1, comparando os trabalhos relacionados e este trabalho.

3.2 Critérios de busca

Para seleção dos trabalhos apresentados, foi realizada uma revisão sistemática ainda durante o começo do projeto de pesquisa. Nesta revisão foram consideradas alguns termos-chave como: junção espacial, junção espacial distribuída, processamento junção espacial, *r-tree*, *spatial join*, *multi-way spatial join*.

Depois de uma rápida analise foram definidos 10 trabalhos relacionados. Devido ao fato de uma nova proposta de algoritmo ser feita, utilizando como base métodos de otimização da *r-tree*, outros 5 trabalhos relacionados foram adicionados. Os trabalhos que mostram-se mais importantes na implementação do novo método estão descritos na Seção 3.4.

3.3 Metodologia de análise

Para a analise dos trabalhos serão adotados 5 critérios. Cada um destes é uma parte fundamental do método em si, pois sem algum deles não seria possível obter os mesmos resultados. Alguns dos trabalhos apresentam em sua composição mais critérios do

que outros, o que fica claro na Tabela 1. Nas subseções a seguir estão especificados cada um dos critérios utilizados e as suas fundamentações.

3.3.1 Proximidade

A localização espacial vai ser utilizada na analise dos conjuntos de dados de uma determinada consulta, e a proximidade será um dos critérios levados em conta para decidir como os alocar os objetos nos servidores disponíveis.

Para a realização das operações de junções espaciais a proximidade dos objetos é um fator de vital importância, pois existe a necessidade de que objetos que se relacionam estejam no mesmo servidor para que possam ser analisados em conjunto.

Embora a maioria dos métodos leve em consideração a proximidade para a distribuição, por ser um fator fundamental, não poderia deixar de ser mencionada nos critérios.

3.3.2 Execução em *Cluster*

A adaptação do algoritmo para utilização de *clusters* de computadores é também um critério fundamental na implementação. Conforme mencionado anteriormente, devido a complexidade e tamanho dos conjuntos de dados, é mais interessante a utilização do processamento paralelo.

Alguns dos métodos apresentados nos trabalhos somente levam em consideração a indexação dos índices dos objetos em uma estrutura de árvore, de modo que os dados de partes especificas do *dataset* sejam acessadas. O intuito do método proposto nesta monografia é o particionamento dos dados de forma a otimizar operações de junção espacial.

3.3.3 Reinserção

Um dos aspectos que torna o método proposto mais eficiente é o fato dele ser compatível a conjuntos de dados dinâmicos, ou seja, caso a ordem ou quantidade dos objetos mude, o algoritmo não terá problemas de tratar o novo conjunto de dados de forma eficiente. Para que isso seja possível, é necessário haver a possibilidade de retirar objetos já inseridos e realoca-los em outros servidores.

Poucos dos trabalhos relacionados contam com algoritmos para tratar a reinserção, que também é um dos critérios fundamentais para o resultado.

3.4 Trabalhos analisados

Esta seção enumera os trabalhos que são considerados os mais relevantes, relativos aos critérios selecionados e apresentados previamente. Alguns dos trabalhos tem presentes mais de um critério, o que os torna mais relevantes para o desenvolvimento do método proposto. Posteriormente serão comparados os trabalhos e métodos propostos em relação ao método *Gain-Loss*.

3.4.1 Análise da Influência do Fator Distribuição Espacial dos Dados no Desempenho de Métodos de Acesso Multidimensionais

A indexação de dados espaciais já foi amplamente trabalhada na literatura, e diversos métodos, estruturas de dados e modificações das mesmas foram propostas para este intuito. Os métodos de acesso são projetados com o intuito de melhorar o desempenho do acesso a objetos espaciais em um sistema de banco de dados.

Em seu trabalho, Ciferri (2002) faz um levantamento e realiza comparações entre o desempenho dos principais métodos de acesso propostos na literatura, entre eles a árvore R* e seus métodos de reinserção, que são de grande importância para que a ordem da leitura dos dados não influencie muito na distribuição. Nos testes, são utilizadas quatro tipos de consultas espaciais.

Ciferri (2002) mostra com seus testes de desempenho que, mesmo sem as alterações propostas para o método *Gain-Loss*, a árvore R* se mostra muito competitiva em relação as outras estruturas. Este fato incentivou melhorias nos métodos desta estrutura e a criação de novos, como no trabalho descrito na Subseção 3.4.2.

3.4.2 *Improving the R*-tree with Outlier Handling Techniques*

O trabalho de Xia e Zhang (2005) foi de grande importância no desenvolvimento do método *Gain-Loss*. Neste artigo, os autores propõem o uso de uma modificação da árvore R* como método de indexação de objetos espaciais.

No seus experimentos, Xia e Zhang (2005) utilizam cinco tipos diferentes de operações espaciais, incluindo junção espacial. O foco dos experimentos é demonstrar um novo tipo de arvore denominada R^0, que contém além de métodos de qualidade, ganho e perda, melhorias na reinserção da R*, de modo a melhor identificar e acomodar objetos considerados atípicos, com muito afastamento dos demais. O método também conta com capacidade de indexamento dinâmico.

A árvore R^0 tem um ótimo desempenho como estrutura de indexação, pois consegue particionar de forma a evitar a sobreposição de MBRs, mesmo em conjuntos de dados complexos.

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

Com enfoque na distribuição de dados espaciais com o uso de *cluster* visando o processamento paralelo, o trabalho de Oliveira et al. (2013) propõe um método de distribuição chamado de *Proximity Area*, que otimiza o processamento de junções espaciais para analise de dados dinâmicos e volumosos.

O *Proximity Area* foi implementado sob um SIG, utilizando a árvore R* sem reinserção. São distribuídos os objetos pelos servidores de acordo com o critério de aumento da área do MBR do servidor, fazendo com que o objeto seja inserido no servidor disponível que resulte no menor aumento. Para a escolha dos servidores que estarão disponíveis para inserção, é determinado pelo usuário antes do inicio da consulta um fator de balanceamento, que limita a diferença na quantidade de objetos que podem ser inseridos nos servidores. O nível de paralelismo da consulta depende do fator de balanceamento escolhido.

Para a utilização do método proposto por Oliveira et al. (2013), foram realizadas algumas adaptações como a retirada da estrutura de árvore para que a consulta fosse processada com particionamento disjunto.

3.4.4 *Efficient Processing of Multiway Spatial Join Queries in Distributed Systems*

Esta tese de Oliveira (2017) é um amplo trabalho na área de multijunções espaciais, que tem como principal contribuição a proposta de uma suíte otimizadora de consultas de multijunção espacial baseada no custo. Este otimizador leva em conta particionamento de objetos, paralelismo da consulta e economia dos recursos do *cluster* para selecionar um bom plano de execução para a consulta.

O trabalho, além de identificar características relevantes no processamento de multijunção espacial, também propõe métodos para calcular o custo do processamento de consultas organizadas por meio de histogramas multidimensionais, no ambiente de *cluster*. Também são propostos métodos para identificação do melhor plano de execução para a consulta, levando em consideração os recursos consumidos e o paralelismo da execução.

Todos os métodos e funcionalidades propostas por Oliveira (2017) foram descritos na publicação em pseudo-código. O trabalho focou nos aspectos mencionados acima e considerou como distribuição somente o método *Round-Robin*, por considerar uma distribuição que apresenta desafios para a etapa de otimização da consulta. No entanto, a avaliação de outras distribuições foi deixada como trabalho futuro.

3.5 Resumo Comparativo

A partir da análise dos trabalhos relacionados é possível observar que cada um deles teve grande contribuição para o resultado final do método *Gain-Loss*.

Primeiramente no trabalho de Ciferri (2002) foi possível visualizar como alguns aspectos da árvore R* produziam ótimos resultados na indexação de objetos espaciais. A partir das melhorias propostas na R* por Xia e Zhang (2005), foi desenvolvido o método *Gain-Loss*, que contou com adaptações na questão do paralelismo da execução, baseadas nos resultados obtidos por Oliveira et al. (2013).

Com base nos critérios definidos na Seção 3.3, foi criada a Tabela 1, que compara os métodos de distribuição alternativos ao *Gain-Loss* e suas limitações em relação aos objetivos esperados com este trabalho. É possível visualizar como cada um dos trabalhos relacionados contribuiu com o desenvolvimento do novo método.

Tabela 1 – Comparativo

	Proximidade	*Cluster*	Reinserção
DGEO	×	√	×
R*-tree	√	×	√
R^0-tree	√	×	√
Proximity Area	√	√	×
Gain-Loss	√	√	√

4 O MÉTODO DE DISTRIBUIÇÃO *GAIN-LOSS*

4.1 Introdução

Neste capítulo será apresentado o novo método de distribuição proposto chamado de *Gain-Loss*, por ter grande influência da árvore R^0 e suas métricas de calculo de qualidade, ganho e perda. As alterações feitas, em sua maioria, foram no intuito de adaptar os métodos da R^0 em um ambiente de *cluster*, para que pudesse ser executado de forma paralela.

4.2 O Método

Apesar da árvore R^0 ser uma estrutura muito boa na indexação de objetos espaciais, conforme apontado anteriormente na Seção 2.2, o processamento dos dados espaciais atualmente tende á utilização de *clusters*, o que não é uma característica intrínseca da árvore. Foram feitas alterações no algoritmo original determinando que os objetos fossem alocados nos servidores indicados, não utilizando mais a estrutura de árvore para indexar os objetos.

Para escolher qual dos servidores é a melhor opção para alocar um novo objeto, o método compara o valor da perda de se inserir o objeto em cada um dos servidores e escolhe o que gera menor perda. A Figura 6 exemplifica com o método aloca os objetos de acordo com a quantidade de servidores disponíveis, primando por gerar MBRs menores e mais quadráticos quanto possível.

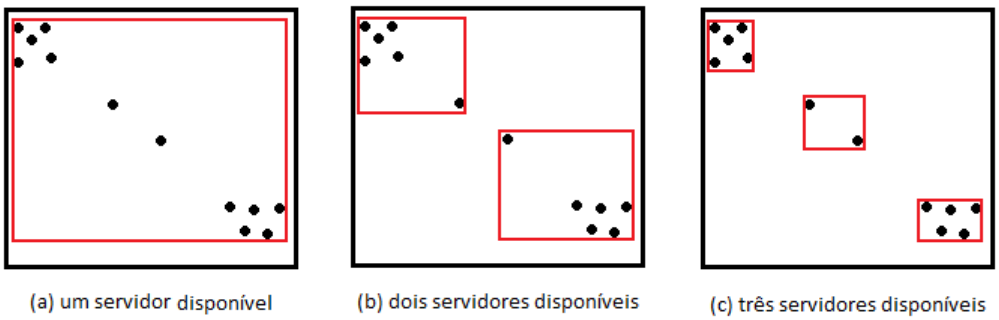

Figura 6 – Divisão dos objetos de acordo com a quantidade de servidores
Fonte: Adaptado de (XIA; ZHANG, 2005)

Com a intenção de manter o paralelismo da execução, ou seja, balancear a quan-

tidade de objetos entre todos os servidores, foi definida uma fórmula demonstrada na Equação 4.1, baseada na quantidade de objetos total dos *datasets* e o total de servidores. A fórmula tem como entrada o número de servidores s e número de objetos n e retorna um teto inteiro M que é a quantidade mínima de objetos que o servidor deve conter para que, em uma nova inserção, seus objetos sejam analisados e realocados em um servidor ainda não utilizado.

$$M = 1 * \left(\left(\frac{\frac{n}{s}}{n - \frac{n}{s}} \right) * n \right) \tag{4.1}$$

Se o servidor escolhido, após a inserção do novo objeto, ainda não conter mais objetos do que o valor de M, não é necessário realizar outra operação. Caso o número de objetos ultrapasse M, o método divide parte dos objetos para um novo servidor. Para realizar a divisão, os objetos são removidos de acordo com a métrica de ganho, definida pela Equação 2.2, avaliando qual conjunto de objetos resultaria em um maior ganho caso fosse removido daquele servidor. O melhor conjunto é escolhido, removido do seu servidor atual e realocado em um servidor vazio.

Caso durante uma inserção for verificado que não existe mais nenhum servidor vazio, o local de alocação será determinado pela métrica de perda, escolhendo o servidor que resultar em um valor menor. Durante esta parte do processo, os servidores não tem mais seus objetos divididos, porém continuam a ser avaliados pois podem ter objetos realocados caso eles se tornem *outliers*.

A técnica *Gain-Loss* é descrita no Algoritmo 4.2, tendo como entrada o número de objetos n, o tamanho do *cluster* s, o MBR do próximo objeto MBR e o servidor atual S. O intuito do algoritmo é receber os dados sobre os servidores e objetos, determinar um servidor onde a alocação resulte em menor custo e alocar o objeto. Caso o servidor fique lotado após a inserção, ele deve ter sua carga dividida com um novo servidor.

São utilizadas algumas funções, como a função $escolheMenorPerda$ que avalia os servidores e escolhe um para o objeto ser inserido, de acordo com a métrica de perda. A função $adcionaObjeto$ vai inserir o objeto da interação atual no servidor determinado. A função $inserçãoForçada$ vai inserir um objeto no servidor determinado mesmo que a quantidade de objetos naquele servidor ultrapasse o valor de M, que é o número máximo de objetos que podem ser inseridos em um único servidor. A última função utilizada é a $splitNoServidor$ que vai dividir os objetos do servidor atual com um novo servidor, ainda não utilizado.

GAIN-LOSS(n, s, MBR, S)

```
1  M ← 1* ( ( n/s / n − n/s ) * n )
2  servidoresUtilizados ← 1
3  for cada objeto n
4       cs ← ESCOLHEMENORPERDA(servidoresUtilizados, S, MBR)
5       if número de objetos em S < M ou servidoresUtilizados = s
6            ADCIONAOBJETO(S, MBR)
7            S ← número de objetos de S + 1
8       else
9            ADCIONAOBJETO(S, MBR)
10           S ← número de objetos de S + 1
11           INSERÇÃOFORÇADA(S, servidoresUtilizados, cs, M)
12           if número de objetos de S > M
13                SPLITNOSERVIDOR(s, S, cs, M, servidoresUtilizados)
14                servidoresUtilizados ← servidoresUtilizados + 1
```

4.3 Experimentos

Para a avaliação dos métodos de distribuição Round-Robin, *Proximity Area* e *Gain-Loss* foram realizados experimentos controlados, para determinar se o comportamento dos métodos ocorreria de forma correta e qual o desempenho dos métodos em diferentes tipos de distribuição de dados e tamanhos de *cluster*.

4.3.1 Métodos de Distribuição

Foram considerados para este trabalho três métodos de distribuição de dados distintos: o método *Round-Robin* (RR); o método *Proximity Area* com três valores de k: 0.1, 0.5 e 0.9; e o método *Gain-Loss*. No *Proximity Area* quando $k = 0.1$, o método força uma quantidade de objetos mais uniforme entre os servidores. O contrário acontece com $k = 0.9$, permitindo que quantidades não-uniformes de objetos sejam distribuídos entre os servidores, porém respeitando a colocalização.

4.3.2 Bases de Dados

Para a execução dos experimentos de avaliação dos métodos, foram construídas três bases de dados que são sintéticas e estão em um espaço com dimensões 100x100. São elas: *i*) uma base uniforme, U, com 500 retângulos distribuídos uniformemente; *ii*) uma base não-uniforme S, com os dados distribuídos de forma concentrada (*skewed*). São 500 retângulos distribuídos usando a lei de Zipf, 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, que representam o MBR dos objetos, variaram de 1 a 10 conforme a distribuição usada para cada base.

4.3.3 *Clusters*

Para a realização dos experimentos, foram considerados cinco tamanhos diferentes de *cluster* disponíveis para se alocar os dados. Foram considerados *clusters* com 4, 8, 16, 32 e 64 servidores. Estes tamanhos foram escolhidos para que fosse representado o desempenho dos métodos tendo poucos computadores disponíveis para distribuição, mas também foi considerado um *cluster* relativamente grande, com 64 computadores. A quantidade de servidores disponíveis para alocar os objetos pode influenciar no balanceamento da consulta.

4.3.4 Configuração

Os testes que foram realizados relacionaram as três bases de dados com os cinco tamanhos de *cluster*, utilizando para a distribuição dos dados os três métodos, descritos nas seções anteriores. A Tabela 2 descreve as configurações utilizadas.

Tabela 2 – Configuração do Experimento

Sigla	*Base de Dados*	*Método de Distribuição*	*Tamanho de Cluster*
C1	Uniforme U	Round-Robin (RR)	4; 8; 16; 32; 64
C2	Uniforme U	*Proximity Area* 0.1 (PA)	4; 8; 16; 32; 64
C3	Uniforme U	*Proximity Area* 0.5 (PA)	4; 8; 16; 32; 64
C4	Uniforme U	*Proximity Area* 0.9 (PA)	4; 8; 16; 32; 64
C5	Uniforme U	*Gain-Loss* (GL)	4; 8; 16; 32; 64
C6	Não-Uniforme S	Round-Robin (RR)	4; 8; 16; 32; 64
C7	Não-Uniforme S	*Proximity Area* 0.1 (PA)	4; 8; 16; 32; 64
C8	Não-Uniforme S	*Proximity Area* 0.5 (PA)	4; 8; 16; 32; 64
C9	Não-Uniforme S	*Proximity Area* 0.9 (PA)	4; 8; 16; 32; 64
C10	Não-Uniforme S	*Gain-Loss* (GL)	4; 8; 16; 32; 64
C11	Combinada C	Round-Robin (RR)	4; 8; 16; 32; 64
C12	Combinada C	*Proximity Area* 0.1 (PA)	4; 8; 16; 32; 64
C13	Combinada C	*Proximity Area* 0.5 (PA)	4; 8; 16; 32; 64
C14	Combinada C	*Proximity Area* 0.9 (PA)	4; 8; 16; 32; 64
C15	Combinada C	*Gain-Loss* (GL)	4; 8; 16; 32; 64

Para cada uma das bases de dados foram determinadas 25 configurações de testes, totalizando 75 configurações distintas de testes que foram realizadas.

4.4 Avaliação dos Resultados

Foram coletados duas métricas de cada uma das configurações de distribuição para avaliar o desempenho dos métodos: sobreposição dos MBRs e desvio padrão do número de objetos em cada servidor após a realização da distribuição. O Algoritmo 4.2 retorna por meio de funções a quantidade de objetos em cada um dos servidores e também a sobreposição total dos MBRs entre eles. O intuito de medir estes dois parâmetros é determinar se, em uma consulta espacial de multijunção, a execução da mesma ocorreria de forma balanceada e com menor uso de rede.

4.4.1 Sobreposição

Para avaliar a qualidade da distribuição dos objetos, foi mensurada 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 4.2, sendo s a quantidade de servidores, $\cup$ representa intersecção, mbr representa o MBR do conjunto de objetos atribuídos ao servidor e $area$ é uma função que calcula a área da interseção entre dois servidores i e j. Uma sobreposição muito grande indica que os objetos não foram agrupados com regiões distintas, o que pode aumentar a cópia de objetos durante a execução da consulta.

$$\eta = \sum_{i=1}^{s-1} \sum_{j=i+1}^{s} area(mbr_i \cup mbr_j) \tag{4.2}$$

A Figura 7 e a Figura 8 apresentm o resultado do experimento para sobreposição de espaço entre os servidores (η). Em (a) tem-se a sobreposição para 4, em (b) para 8, em (c) para 16, em (d) para 32 e em (e) para 64 servidores.

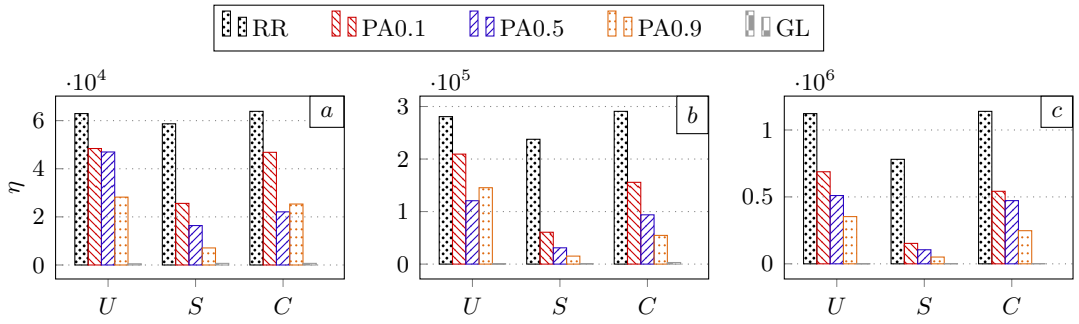

Figura 7 – Sobreposição espacial entre os MBRs dos servidores. (a) mostra a sobreposição para 4 servidores, (b) para 8 servidores e (c) para 16 servidores.

Há um padrão nos gráficos (a), (b) e (c) indicando que a sobreposição descresce 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 barra no gráfico.

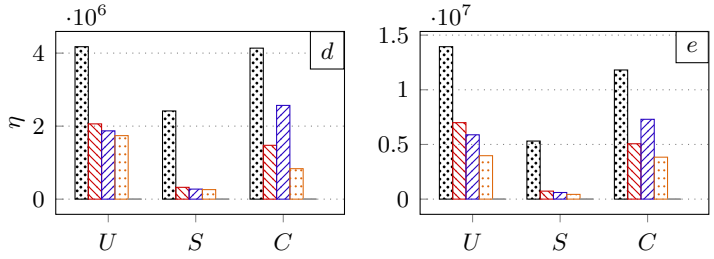

Figura 8 – Sobreposição espacial entre os MBRs dos servidores. (*d*) mostra a sobreposição para 32 servidores e (*e*) para 64 servidores.

Nos gráficos (d) e (e) é possível observar o mesmo padrão de decrescimento observado nos gráficos anteriores, porém na base C o PA 0.5 tem um desempenho inferior ao PA 0.1 e PA 0.9. Novamente o método RR aparece com maior sobreposição, indicando um pior desempenho, e o método GL mostrou-se novamente muito superior aos demais.

4.4.2 Desvio Padrão

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, necessitará copiar dados de outros servidores.

A Figura 9 e a Figura 10 apresentam o desvio padrão, σ, da população de quantidades de objetos em cada servidor.

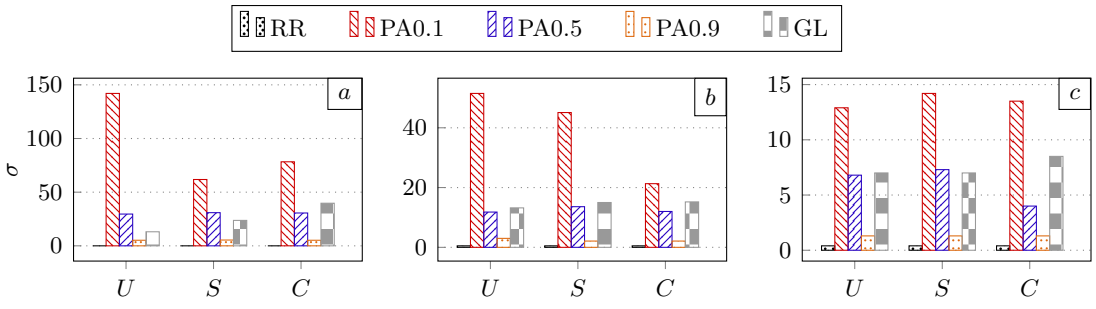

Figura 9 – Desvio da população de quantidade de objetos em cada servidor conforme o método de distribuição. (*a*) mostra o desvio para 4 servidores, (*b*) para 8 servidores e (*c*) para 16 servidores.

Conforme observado nos gráficos (a), (b) e (c), em questão de desvio padrão, o método PA 0.1 é o mais desbalanceado dentre todos os outros métodos. O método PA 0.9 tem um comportamento muito semelhante ao RR em relação ao desvio padrão de

população. É possível observar também que o método GL se mantem próximo ao PA 0.5, tendo um pior cenário com o um *cluster* de tamanho 16.

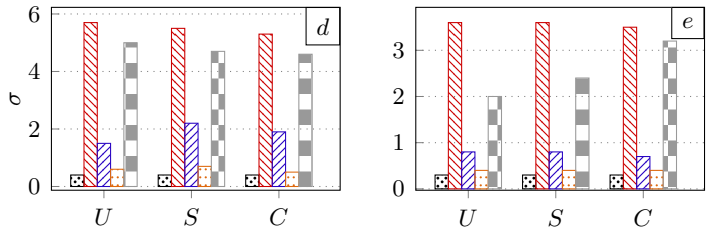

Figura 10 – Desvio da população de quantidade de objetos em cada servidor conforme o método de distribuição. (*d*) mostra o desvio para 32 servidores e (*e*) para 64 servidores.

Já nos gráficos (d) e (e) se observa um comportamento ligeiramente diferente do método GL, ficando bem próximo do PA 0.1 em questão de balanceamento, porém a diferença destes dois métodos para os demais diminui. O desempenho dos outros métodos PA 0.5, PA 0.9 e RR permanece igual.

4.5 Considerações Finais

O método *Round-Robin*, em relação ao consumo de rede, teve o pior desempenho. Isso se deve ao fato dele não considerar a colocalização dos objetos para realizar a sua distribuição, tendo alta sobreposição em todo cenário testado. Já na questão do paralelismo, o método é o mais eficiente, dividindo a carga de objetos da forma mais uniforme possível.

O método *Proximity Area* e suas variações ($k = 0.1; 0.5; 0.9$) tiveram um desempenho parecido em todos os tamanhos de *cluster*, exceto no conjunto de dados combinado, onde o comportamento variou para PA 0.5. Em alguns dos cenários testados o *Proximity Area* teve uma sobreposição bem menor em relação ao *Round-Robin*, mas não em todos, indicando que o método ainda não é ideal para colocalizar os objetos de forma adequada. Na questão do paralelismo, as variações de k 0.5 e 0.9 tem um desempenho muito bom, enquanto 0.1 se mantem ligeiramente mais alto. O método *Proximity Area* varia seu desempenho de acordo com o conjunto de dados e com o fator de balanceamento k adotado.

Já o método proposto, *Gain-Loss*, mostrou-se extremamente eficiente para distribuir os objetos de forma a evitar a sua sobreposição desnecessária. Em todos os cenários de teste, ele teve uma sobreposição muito baixa em relação a todos os outros métodos. Isso indica que ele segue um caminho muito bom quanto ao uso de rede, fazendo com que ele seja mínimo. Porém, quando compara-se o desvio padrão da quantidade de objetos nos servidores, o método *Gain-Loss* tem um desempenho ligeiramente baixo em *clusters* maiores, principalmente no *cluster* de tamanho 32, e no conjunto de dados combinado com

cluster de 64 servidores. Embora este desempenho não torne os resultados da execução ruins, mostra que o método ainda pode sofrer adaptações nas equações e métodos utilizados para o controle do paralelismo.

Os conjuntos de dados utilizados para a realização dos experimentos e avaliação dos métodos são sintéticos, e as métricas utilizadas, mesmo que sejam fatores de grande influência nas consultas de multijunção espacial, não representam exatamente a execução de uma consulta real, onde seria avaliado o uso de rede e o tempo de execução da consulta para medir o desempenho dos métodos.

5 CONCLUSÕES E TRABALHOS FUTUROS

5.1 Introdução

Neste capítulo serão abordadas as conclusões tiradas a partir da análise dos resultados do experimento. Foram comparados três métodos de distribuição de dados em sistemas distribuídos, sendo um deles com 3 valores diferentes de balanceamento, afim de identificar qual deles tem melhor desempenho. As métricas usadas para avaliar os métodos foram sobreposição e desvio padrão, que indicam uma execução com menos uso de rede e mais balanceada, caso os valores das métricas sejam baixos.

5.2 Conclusões

Dentre os três métodos apresentados e testados, no quesito consumo de rede, o método *Round-Robin* foi o menos efetivo, seguido do *Proximity Area* e suas variações, sendo então o método *Gain-Loss* o melhor neste quesito, pois em todos os cenários obteve valores de sobreposição extremamente baixos em relação aos outros métodos avaliados.

Em relação a distribuição balanceada dos objetos, como esperado, o método *Round-Robin* foi muito mais efetivo comparativamente aos outros dois métodos. Sua forma de distribuição força o balanceamento ao máximo. Os métodos *Proximity-Area*, em algumas variações, e o *Gain-Loss* obtiveram resultados parecidos em relação ao balanceamento da distribuição, sendo que nas configurações onde o*Proximity-Area* apresenta um valor de desvio padrão mais próximo do *Gain-Loss*, a sobreposição do primeiro é grande.

O novo método proposto, apesar de ter um desempenho satisfatório nos testes, ainda pode passar por mudanças e adaptações para que se torne mais eficiente no quesito balanceamento.

5.3 Trabalhos futuros

Embora a pesquisa tenha proporcionado resultados interessantes quanto ao desempenho dos métodos de distribuição de dados em diferentes tamanhos de *cluster* e tipos de base de dados, em um trabalho futuro seria interessante utilizar estes métodos no processamento de consultas de multijunção espacial com conjuntos de dados reais, o que não foi possível devido ao tempo desprendido para o desenvolvimento do método

Gain-Loss. É esperado que a avaliação do tempo de execução das consultas e do de rede vá de encontro com os dados encontrados nesta pesquisa.

Outro trabalho futuro possível é o aprimoramento dos métodos utilizados pelo *Gain-Loss* para manter o balanceamento da distribuição. Podem ser alterados alguns fatores, de modo a tornar a distribuição dos objetos mais homogênea, aumentando o aproveitamento do *cluster* na execução da consulta. Mesmo que alterações resultem em maior sobreposição, isso pode ser aceitável até certo ponto, pois o método conta com uma sobreposição extremamente baixa em relação aos outros.

O aspecto do balanceamento não ser o ideal foi identificado durante a realização dos experimentos. Experimentos com dados reais também eram uma expectativa, mas tais atividades foram retiradas do escopo do trabalho devido ao curto prazo de execução e também devem ser consideradas em trabalhos futuros.

REFERÊNCIAS

BACELLAR, H. V. Cluster: Computação de alto desempenho. *Instituto de Computação, Universidade Estadual de Campinas. Campinas, São Paulo*, 2009. Citado 3 vezes nas páginas 12, 13 e 21.

BECKMANN, N. et al. The R*-tree: an efficient and robust access method for points and rectangles. v. 19, n. 2, p. 322–331, 1990. Citado na página 19.

CIFERRI, R. R. Análise da influência do fator distribuição espacial dos dados no desempenho de métodos de acesso multidimensionais. Universidade Federal de Pernambuco, 2002. Citado 6 vezes nas páginas 12, 18, 19, 21, 32 e 34.

CUNHA, A. R. et al. Processamento distribuído da junção espacial de múltiplas bases de dados: multi-way spatial join. Universidade Federal de Goiás, 2014. Citado na página 26.

FITZ, P. R. *Geoprocessamento sem complicação*. [S.l.]: Oficina de textos, 2018. Citado 4 vezes nas páginas 15, 16, 17 e 18.

GUTTMAN, A. R-trees: A Dynamic Index Structure for Spatial Searching. v. 14, n. 2, p. 47–57, 1984. Citado na página 19.

MAMOULIS, N.; PAPADIAS, D. Multiway spatial joins. *ACM Transactions on Database Systems (TODS)*, ACM, v. 26, n. 4, p. 424–475, 2001. Citado 2 vezes nas páginas 25 e 27.

OLIVEIRA, S. de et al. Processamento distribuído de operações de junção espacial com bases de dados dinâmicas para análise de informações geográficas. *XXXI Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos*, 2013. Citado 6 vezes nas páginas 12, 20, 23, 24, 33 e 34.

OLIVEIRA, T. B. de. *Efficient Processing of Multiway Spatial Join Queries in Distributed Systems*. Tese (Doutorado) — Instituto de Informática, Universidade Federal de Goiás, Goiânia, GO, Brasil, 11 2017. Citado 14 vezes nas páginas 12, 15, 16, 17, 18, 20, 22, 23, 24, 26, 27, 28, 29 e 33.

OLIVEIRA, T. B. de; COSTA, F. M.; RODRIGUES, V. J. S. Definição de Planos de Execução Distribuídos para Consultas de Junção Espacial usando Histogramas Multidimensionais. In: . Petrópolis, RJ, Brasil: [s.n.], 2015. p. 89–100. Citado 3 vezes nas páginas 12, 27 e 28.

ÖZSU, M. T.; VALDURIEZ, P. *Principles of distributed database systems*. [S.l.]: Springer Science & Business Media, 2011. Citado na página 29.

PATEL, J. M.; DEWITT, D. J. Clone join and shadow join: two parallel spatial join algorithms. In: ACM. *Proceedings of the 8th ACM international symposium on Advances in geographic information systems*. [S.l.], 2000. p. 54–61. Citado 2 vezes nas páginas 12 e 23.

ROH, Y. J. et al. Hierarchically organized skew-tolerant histograms for geographic data objects. In: ACM. *Proceedings of the 2010 ACM SIGMOD International Conference on Management of data*. [S.l.], 2010. p. 627–638. Citado 2 vezes nas páginas 22 e 27.

TANENBAUM, A. S.; STEEN, M. V. *Distributed systems: principles and paradigms*. [S.l.]: Prentice-Hall, 2007. Citado na página 21.

XIA, T.; ZHANG, D. Improving the r*-tree with outlier handling techniques. In: ACM. *Proceedings of the 13th annual ACM international workshop on Geographic information systems*. [S.l.], 2005. p. 125–134. Citado 5 vezes nas páginas 19, 21, 32, 34 e 35.