DSI-RTree - Um Índice R-Tree Escalável Distribuído

Thiago B. de Oliveira[1], Vagner J. do Sacramento Rodrigues[1], Sávio S. T. de Oliveira[1], Pedro I. de Albuquerque Lima[1], Marcelo de C. Cardoso[1]

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

{thborges,vsacramento,savioteles,pedroivanlima,marcelodcc}@gmail.com

Abstract. *This work presents a distributed and scalable system called DSI-RTree, which implements a distributed index to process spatial data in a cluster of computers. Issues such as the size of data partitions, how that partitions are distributed and the impact of these choices in the message flow on the cluster are reviewed. An equation to calculate the size of the partitions is proposed. We have done experiments running window queries in spatial data sets of 33,000 and 158,000 polygons and the results showed a scalability greater than linear.*

Resumo. *Este trabalho apresenta o DSI-RTree, um sistema distribuído e escalável, que implementa a indexação e processamento distribuído de dados espaciais em um* cluster *de computadores. O tamanho das partições criadas, a forma de distribuição destas partições e o impacto destas definições na troca de mensagens entre as máquinas do* cluster *são discutidos. Uma fórmula para cálculo do tamanho das partições é proposta. Testes práticos do sistema mostraram uma escalabilidade maior que linear no processamento de consultas de janela em datasets[1] espaciais de 32 e 158 mil polígonos.*

1. Introdução

Certamente estamos na era do mundo ubíquo geográfico alinhado à visão de Mark Weiser [Weiser 1995], devido a grande disponibilidade de dados espaciais e os celulares com GPS/GPRS cada vez mais comuns e acessíveis. Entretanto, muitos desafios ainda precisam ser tratados para prover aplicações LBS (*Location Based Services*) em larga escala. Um deles é a falta de escalabilidade das plataformas de geoprocessamento.

Uma aplicação LBS, usada por milhões de usuários, necessita processar milhares de requisições em tempo real, usando a localização geográfica de cada usuário para encontrar possíveis pontos de interesse colocalizados de acordo com suas preferências.

Clusters de máquinas convencionais possuem um custo reduzido em relação a máquinas paralelas de alto desempenho, além de outras facilidades para prover escalabilidade e disponibilidade a um sistema SIG [Bernhardsen 2002]: i) Os dados espaciais podem ser distribuídos pelas máquinas do *cluster*, de forma que cada uma armazene e processe requisições em parte do dataset; ii) É possível aumentar a capacidade de armazenamento e processamento, adicionando ou removendo máquinas ao *cluster* existente, tornando o sistema escalável horizontalmente; iii) Torna o investimento, em *hardware* e energia, proporcional ao aumento da demanda pelo serviço.

[1]O termo dataset foi empregado no texto para se referir a um conjunto de dados espaciais, como por exemplo um arquivo *shapefile*. Do Inglês *data set*.

Porém, existem vários desafios na implementação de soluções eficientes para armazenar e processar grande volume de dados espaciais de forma distribuída. Métodos de divisão devem ser empregados para possibilitar o armazenamento de partes dos dados (partições) em cada uma das máquinas de um *cluster*, de forma a possibilitar a divisão do problema (algoritmo) a ser processado em paralelo de forma eficiente. À medida que mais máquinas são empregadas no cálculo da solução, devem ser considerados ainda o uso uniforme do *cluster* e a necessidade de reduzir a comunicação.

Neste trabalho foi projetado e implementado um sistema de processamento distribuído de dados espaciais, chamado DSI-Rtree (Distributed Spatial Index-Rtree), que particiona e distribui os dados, a fim de prover escalabilidade no armazenamento e processamento de consultas espaciais em arquitetura de computadores distribuída (*clusters*).

O restante do trabalho está organizado da seguinte forma: A Seção 2 faz uma revisão da literatura sobre estruturas e algoritmos propostos para indexação de dados espaciais; a Seção 3 apresenta a arquitetura e os detalhes de implementação; a Seção 4 apresenta a forma de definição do tamanho das partições dos dados na arquitetura proposta; a Seção 5 apresenta os resultados dos testes de desempenho; a Seção 6 discute os trabalhos correlatos e, por fim, a Seção 7 apresenta a conclusão e trabalhos futuros.

2. Estruturas de Dados para Processamento de Dados Espaciais

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

A R-Tree é uma árvore balanceada por altura, semelhante à B^+-Tree [Comer 1979], com ponteiros para objetos espaciais nos nós folhas. É uma estrutura de dados hierárquica que utiliza retângulos (chamados de MBRs - *Minimum Bounding Rectangle*) para organizar um conjunto dinâmico de objetos espaciais, de maneira que objetos colocalizados fiquem armazenados próximos uns dos outros e que haja uma redução no espaço de busca a cada nível da árvore [Guttman 1984].

Entre as várias extensões propostas para a R-Tree, a R*-Tree [An et al. 2003] foi escolhida para implementação da arquitetura proposta neste trabalho. Esta variante propõe mecanismos para melhorar o tempo de busca [Beckmann et al. 1990] e é implementada por vários bancos de dados espaciais, tais como PostGis e Oracle Spatial.

Conforme ilustrado na Figura 1, a estrutura hierárquica da R-Tree possui um nó raiz, nós internos (N1..2 $\subset$ N3..6) e um último nível de nós folha (N3..6 $\subset a..i$). A parte superior da figura retrata o desenho dos MBRs agrupando os objetos espaciais de a a i em subconjuntos, de acordo com sua colocalização. A parte inferior ilustra a representação da árvore R-Tree. Cada nó armazena no máximo M e no mínimo $m \leq \frac{M}{2}$ entradas [Guttman 1984]. O valor de M é um fator importante na organização de um índice R-Tree. A escolha deste valor pode gerar mais espaço morto e áreas sobrepostas entre os MBRs, o que degrada o desempenho da operação de busca na árvore.

Espaço morto é a área adicional do MBR necessária para cobrir o polígono como um todo. Na parte superior da Figura 1, a área de N1 não preenchida pelas suas entradas é um exemplo de espaço morto. Áreas sobrepostas são regiões de interseção entre polígonos. Um exemplo de sobreposição é ilustrado na Figura 1, entre N3 e N4.

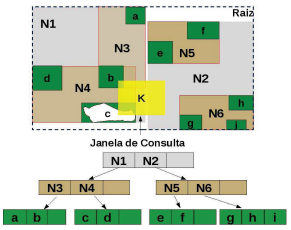

Figura 1. Organização de uma R-Tree.

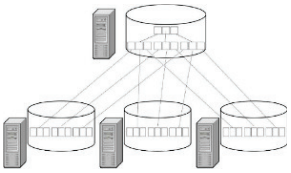

Figura 2. Particionamento de uma R-Tree [Manolopoulos et al. 2006].

A Consulta de Janela é um dos principais algoritmos de busca na R-Tree. Seu objetivo é encontrar todos os objetos espaciais dentro de uma determinada área, que é especificada através de uma janela (retângulo K na Figura 1). O algoritmo inicia pelo nó raiz e percorre a árvore comparando a janela com o MBR de cada entrada dos nós internos, seguindo nos ramos onde existe interseção. Nas folhas, a janela de consulta é comparada com cada objeto espacial do nó. São retornados os itens da folha que contém interseção com a janela de consulta. A janela de consulta K só irá retornar os itens b e c.

O espaço morto causa o caminhamento em subárvores falsas, que não possuem objetos espaciais que intersectam a janela. Na Figura 1, a janela K apresenta interseção com o espaço morto do MBR de N2 e, por isso, a consulta segue pela subárvore de N2, apesar dela não conter nenhum item a ser retornado. A redução de áreas sobrepostas evita que subárvores sejam acessadas desnecessariamente durante o caminhamento na estrutura. Na Figura 1, a área de sobreposição entre N3 e N4 obriga o algoritmo a processar as duas subárvores, o que degrada o desempenho da R-Tree [Guttman 1984, Beckmann et al. 1990].

2.1. Particionamento da R-Tree em Arquiteturas *Shared-Nothing*

Para processar dados em arquiteturas *shared-nothing* (*cluster*) [DeWitt and Gray 1992] dois requisitos principais devem ser implementados por um sistema computacional: i) a divisão dos dados em partições; e ii) a distribuição destas partições entre as máquinas do *cluster*. Para datasets espaciais, esta divisão, também chamada de particionamento, e a distribuição das partições consideram a colocalização geográfica entre os polígonos para acelerar o posterior processamento de algoritmos de busca.

A Figura 2 ilustra a estrutura lógica geral de uma R-Tree distribuída em um *cluster* de computadores. O particionamento dos dados é realizado agrupando os nós da árvore em servidores do *cluster*, de forma indexada segundo a estrutura lógica de uma R-Tree. As linhas na figura representam a necessidade de troca de mensagens para alcançar as subárvores durante o processamento.

Os algoritmos de inserção, ajustes e as buscas efetuadas na R-Tree distribuída podem ser executados de forma semelhante ao que é realizado na versão centralizada, exceto pela i) necessidade de troca de mensagens para acessar as partições distribuídas e ii) pelo tratamento de concorrência e consistência necessário devido ao processamento paralelo dos algoritmos.

A divisão dos dados em partições distribuídas no *cluster*, apesar de ser o principal fator que proporciona o processamento paralelo dos algoritmos, também causa o principal desafio da construção do índice distribuído: a comunicação necessária entre as máquinas que armazenam as partições. A comunicação também é influenciada pela qualidade do

índice. Quando bem construído o índice reduz o espaço de busca e o acesso a subárvores, o que consequentemente diminui o número de mensagens trocadas.

3. Arquitetura e Implementação do DSI-RTree

Nesta seção são apresentados destalhes da arquitetura e implementação que caracterizam os desafios na contrução de uma solução para geoprocessamento distribuído. Conforme a taxonomia definida em [An et al. 1999], o índice distribuído foi construído com as seguintes características: i) **Unidade de alocação**: bloco – para cada nó da R-Tree é criada uma partição; ii) **Frequência de alocação**: *overflow* – novas partições são criadas durante a inserção quando há uma divisão de um nó da árvore; iii) **Política de distribuição**: balanceada – as partições são distribuídas pelas máquinas do *cluster* de forma a manter o balanceamento dos dados.

Uma visão geral da arquitetura distribuída é ilustrada na Figura 3. Existem três camadas na arquitetura: i) A **Camada de Aplicações**, composta pela API Cliente e Aplicações que usam o serviço. Tem a função de abstrair os detalhes de conexão e comunicação com as demais camadas e, com isso, facilitar a construção das aplicações. A API disponibiliza métodos para criação e atualização de índices de datasets espaciais, bem como métodos para a execução de consultas nos dados armazenados no serviço; ii) A **Camada de Armazenamento e Processamento**, composta pelos Servidores DSI (N1, N2.. Nn) e o Distribuidor de Partições. Os Servidores DSI ficam distribuídos nas máquinas do *cluster* – um em cada máquina – e são os responsáveis pelo armazenamento e processamento dos dados geoespaciais. Neles são armazenadas as partições do índice e executados os algoritmos da R-Tree. A decisão do local de armazenamento das partições é efetuada pelo componente *Distribuidor*. Durante a construção do índice, sempre que ocorre a divisão de uma partição, o Distribuidor é consultado para determinar qual será o local de armazenamento da nova partição criada; iii) A **Camada de Comunicação**, que tem a função de i) intermediar as requisições da API Cliente e repassá-las para os Servidores DSI adequados, ii) prover um mecanismo de comunicação entre os Servidores DSI e iii) disponibilizar o mecanismo de comunicação necessário para a entrega distribuída do resultado do processamento das consultas requisitadas pela API Cliente. É implementada por um *Middleware* Orientado a Mensagens (MOM - *Message Oriented Middleware*) escalável e tolerante a falhas que mantém canais de comunicação – tópicos – com cada um dos Servidores DSI. O envio de mensagens através dos tópicos utiliza o mecanismo de Subscrição e Publicação (*Publish/Subscribe*) do MOM.

Além da comunicação entre os Servidores DSI, existem outros dois componentes na Camada de Comunicação: o Diretório de Registro e o Barramento de Entregas. O

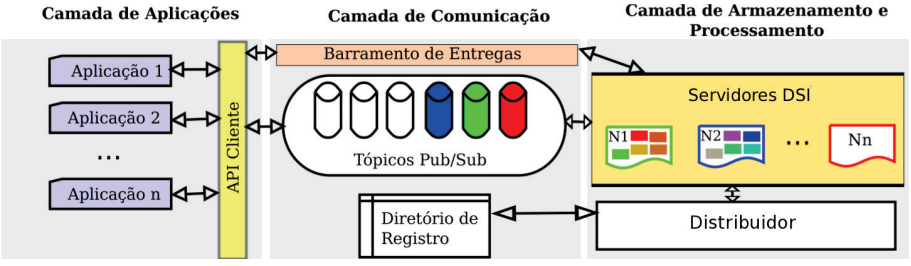

Figura 3. Visão Geral da Arquitetura do DSI-RTree

Diretório de Registro contém a lista de Servidores DSI conectados ao serviço, a qual é consultada pelo Distribuidor para determinar quais Servidores DSI estão disponíveis para armazenamento de partições. O Barramento de Entregas é o canal de comunicação para a entrega de resultados de consultas requisitadas pela API Cliente. Por ele trafegam os polígonos dos datasets que coincidem com as buscas requisitadas.

Uma decisão arquitetural importante do DSI-RTree é a estruturação e o armazenamento dos objetos espaciais e seus atributos em RAM. Ele aproveita não somente o poder de processamento do *cluster*, que é escalável horizontalmente, mas também a memória RAM disponível. Em geral, os bancos de dados geoespaciais existentes mantêm os dados em memória secundária (disco rígido) e fazem uso de cache em RAM para tornar mais eficiente o acesso aos dados. Porém, a paginação em disco degrada o desempenho do sistema devido ao custo de I/O em grandes bases de dados. O DSI-RTree também persiste os objetos espaciais em disco de forma descentralizada, fornecendo um mecanismo de recuperação em caso de necessidade de manutenção no *cluster*.

A construção e os testes de uma infraestrutura distribuída para processamento de objetos espaciais é complexa. Para facilitar e dar suporte à construção, e também medir o desempenho em função dos objetivos de escalabilidade, alguns módulos adicionais foram desenvolvidos: uma plataforma de monitoramento – permite coletar a avaliar informações de *hardware*/*software* do *cluster*; um configurador automático – permite configurar o *cluster* em função da execução do DSI-RTree; um testador automático – permite configurar testes em função do dataset, da quantidade de máquinas e das janelas de consulta.

3.1. Concorrência na Construção do Índice Distribuído

Em uma arquitetura distribuída para indexação de dados espaciais é fundamental reduzir o tempo de construção do índice. Uma forma de alcançar este objetivo é realizar a inserção das geometrias dos datasets de forma paralela entre as máquinas do *cluster*. O tratamento de concorrência é necessário para garantir que o índice contruído seja consistente.

Os algoritmos de construção da R-Tree realizam ajustes nos níveis superiores, de acordo com as mudanças na estrutura que ocorrem nos níveis inferiores. É o caso do algoritmo de divisão de nós do processo de inserção. Quando um nó no nível inferior do índice é dividido, é necessário comunicar o nó pai a respeito das modificações ocorridas em seu MBR e da nova partição resultante da divisão. Este nó pai não deve processar outras requisições na estrutura porque pode tomar decisões considerando um estado inconsistente da estrutura do índice [Chen et al. 1997, Kanth et al. 2002].

Neste trabalho foi implementado a técnica de *lock coupling* conforme descrita em [Chen et al. 1997]. Durante a inserção de uma geometria no índice distribuído, sempre que a raiz de qualquer subárvore está totalmente preenchida, um bloqueio é colocado em seu nó pai. Toda a subárvore é então processada sequencialmente, sendo ajustanda de forma *bottom-up*, da mesma forma realizada pelo algoritmo de divisão das R-Trees.

Os bloqueios são removidos à medida que os ajustes vão sendo realizados, até alcançar o nó raiz da subárvore. A construção em paralelo da subárvore continua após todos os bloqueios serem removidos. Apesar dos bloqueios, subárvores diferentes são construídas em paralelo. A única situação de bloqueio total na estrutura ocorre quando o nó raiz da árvore está prestes a ser dividido.

3.2. Ajuste do MBR *Top-Down*

Uma modificação foi realizada no algoritmo de inserção da R-Tree em relação ao ajuste de MBRs. No algoritmo original, este ajuste é realizado depois que o item é acomodado em um nó folha, de forma *bottom-up* na estrutura. Em nossa implementação, este ajuste foi realizado no momento da escolha da subárvore na qual o item será inserido (*top-down*), conforme discutido por [Chen et al. 1997]. Para plataforma distribuída, esta forma de ajuste é ainda mais importante do que em plataformas paralelas, pois, além de evitar o bloqueio que seria necessário para efetuar este ajuste em todo o ramo acessado, não é necessário o envio de mensagens para notificar os ajustes nos níveis superiores.

Ajustando o MBR antecipadamente, o Servidor DSI pode tomar as mesmas decisões que tomaria se fosse efetuado o bloqueio, no que diz respeito à escolha de subárvores para as próximas mensagens a serem processadas. Isto reduz a quantidade de mensagens na rede e otimiza o processo de construção de um índice espacial distribuído.

3.3. Distribuidor de Partições

O Distribuidor é um componente flexível na arquitetura e pode ser expandido para explorar o maior paralelismo da infraestrutura. Ele é determinante na quantidade de mensagens trocadas e, consequentemente, no desempenho do processamento de consultas. Neste trabalho foi implementado um distribuidor com o propósito de processamento de consultas de janela seletivas, utilizando o método de distribuição de partições *Round-Robin*. A implementação foi realizada de forma distribuída, ou seja, cada Servidor DSI possui uma instância do distribuidor, evitando que i) haja um ponto único de falha na arquitetura e ii) que ocorra o acesso sequencial ao mesmo pelas várias instâncias de Servidores DSI.

O Distribuidor é executado por *overflow*. Uma partição é dividida quando o algoritmo de inserção verifica que a quantidade de itens no nó é igual a M. Para decidir onde colocar a nova partição, o Distribuidor executa os seguintes passos: i) Obtém a lista atualizada de servidores no Diretório de Registro de Servidores DSI; ii) Encontra o servidor onde a nova partição será armazenada, observando a posição do último servidor utilizado e considerando que a lista é circular; iii) Registra o último servidor utilizado.

Para manter um comportamento similar ao de um distribuidor centralizado, após obter a lista de servidores disponíveis do Diretório de Registro pela primeira vez, cada distribuidor encontra a sua própria posição nesta lista, e começa a iterar a partir desta posição. A primeira partição é colocada no próprio Servidor DSI. Desta maneira, mesmo existindo várias instâncias do distribuidor, as partições continuam balanceadas entre as máquinas do *cluster*, de forma semelhante ao que aconteceria se existisse uma só instância do mesmo.

O algoritmo *Round-Robin* foi escolhido por ser facilmente implementado em uma infraestrutura distribuída e apresenta bons resultados para consultas de janela seletivas, como demonstrado na seção 5. Entretanto, não é uma boa opção para operações de join espacial entre datasets diferentes, necessária em muitas aplicações GIS. O distribuidor *Round-Robin* não leva em consideração a colocalização de objetos espaciais entre dois ou mais datasets. Isso permite que objetos espaciais de datasets diferentes, porém relacionados, fiquem em máquinas diferentes, gerando muita troca de mensagem para determinar a relação. O distribuidor é um componente plugável e possibilita que novos algoritmos sejam implementados para atender a esta necessidade.

4. Definição do Tamanho das Partições do Índice Distribuído

Em todos os artigos revisados, a discussão do tamanho das partições (valor de M), quando realizada, é feita em relação ao armazenamento paginado do índice em disco. A arquitetura proposta não utiliza paginação no armazenamento do índice, mantento-o em RAM. A discussão a seguir é focada no desempenho do índice distribuído e na troca de mensagens entre as máquinas do *cluster*, algo ainda não discutido em trabalhos correlatos.

Observando o comportamento logarítmico-recursivo das árvores R em geral, nota-se algumas relações entre o valor de M e o desempenho do índice distribuído. De forma geral, valores de M muito pequenos causam uma árvore com muitos níveis, que necessitam de muita troca de mensagem devido à quantidade de partições. Causam ainda aumento nos MBRs e espaço morto nos níveis superiores, provocando o acesso falso a nós do índice (nós que não contêm resultado para as buscas). Por outro lado, valor de M muito alto, apesar de reduzir a quantidade de mensagens trocadas, pode causar índices sem seletividade alguma (com apenas um nível) ou índices com mais de um nível, porém com nó raiz pouco preenchido e, consequentemente, pouco seletivo.

Para observar estes detalhes, uma avaliação foi realizada com os datasets descritos a seguir e ilustrados na Figura 4, a fim de mensurar o impacto do valor de M na estrutura do índice e também na quantidade de mensagens trocadas durante o processamento de buscas de janelas. Os datasets utilizados nos experimentos são dados reais com polígonos complexos de diferentes regiões: i) **Arizona TabBlock**: extraído do banco de dados de censo demográfico dos Estados Unidos (TIGER 2008), com 158.292 polígonos representando áreas delimitadas (quarteirões e propriedades rurais) do estado do Arizona. Este dataset é referenciado no texto e figuras a seguir como *arizona*; ii) **Alertas de Desmatamento**: dados do Instituto de Geografia da UFG, com 32.578 polígonos, representando alertas de desmatamento no Cerrado brasileiro. É referenciado no texto e figuras a seguir como *desmata*; iii) **Vegetação**: Dados nacionais com 2.140 polígonos representando as áreas de vegetação brasileiras. Referenciado no texto e figuras a seguir como *vegeta*.

Foram construídos índices para os três datasets e executadas buscas com as janelas ilustradas na Figura 4. As métricas coletadas são apresentadas no gráfico da Figura 5. Ele está dividido em três partes: à esquerda o dataset arizona, no meio o dataset desmata e, à direita, a análise do dataset vegeta, com uma maior quantidade de valores de M.

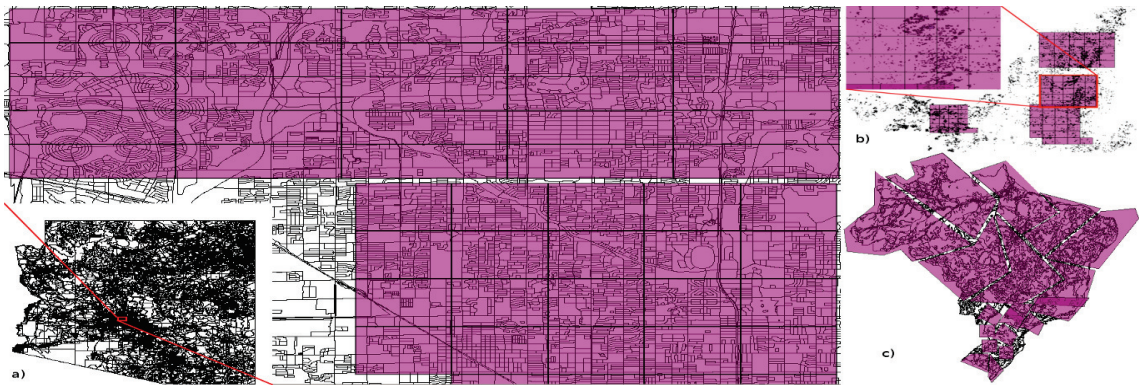

Figura 4. Ilustração dos Datasets e Janelas de Busca: Arizona no canto inferior esquerdo, com ampliação ao seu redor (a), Desmata (b) com ampliação no canto superior direito e Vegeta (c). As áreas em destaque são Janelas de Busca.

Observa-se em todos os datasets que a quantidade de mensagens trocadas diminui, conforme o valor de M aumenta. Para determinados valores de M (Arizona: M50->80, Desmata M=38->42 e Vegeta M17->18 e M50->80) há uma mudança de altura da R-Tree. No entanto, esta mudança não causa um comportamento diferenciado nas linhas do gráfico da Figura 5. A quantidade de mensagens trocadas continua diminuindo, com pequenas oscilações, para todos os valores de M testados. Isto sugere que o fator mais determinante para a redução na troca de mensagens é a diminuição da quantidade de nós no índice, devido à maior capacidade de armazenamento que o valor de M proporciona.

As outras duas linhas no gráfico evidenciam a redução no percentual médio de interseção de itens por nó. Para cada nó do índice, calculou-se a quantidade de itens que são intersectados pela janela, em relação à quantidade de itens existente. O percentual no gráfico é a média de todos os valores obtidos, para nós diretórios e folhas separadamente.

Nota-se uma redução contínua do percentual de itens intersectados por nó, à medida que o valor de M aumenta. Apesar da janela de busca intersectar o MBR do nó, cada vez menos itens contidos no mesmo intersectam com a janela de busca. Para atender o requisito de balanceamento das árvores R (40% mínimo na R*), o nó agrupa itens não tão colocalizados no espaço geoespacial, e isso provoca um aumento de seu MBR.

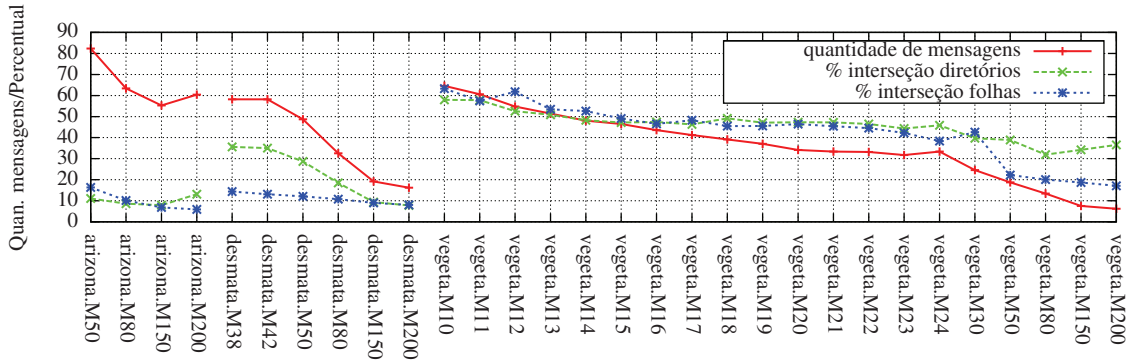

Figura 5. Análise de Métricas em cada Dataset para valores de M variados.

Observou-se ainda que a definição fixa do valor de M, como feito na literatura revisada, provoca uma variação no preenchimento do nó raiz do índice. Dependendo do valor fixo de M e do tamanho do dataset indexado, o nó raiz pode ficar muito ou pouco preenchido. Como a primeira redução do espaço de busca ocorre neste nó, é importante que ele esteja bem preenchido para isolar a maior parte do espaço de busca já na primeira iteração do algoritmo de busca, evitando troca de mensagens desnecessária. Indexar datasets pequenos ou grandes, com valor de M fixo, causa uma baixa seletividade no nó raiz do índice. Se o valor de M é grande, pode gerar índices muito baixos para datasets grandes (pouca seletividade); se é pequeno, gera índices muito altos para datasets pequenos (muita comunicação devido a altura do índice).

Uma possível forma de evitar este comportamento é calcular um valor de M específico para cada dataset. Dado que na arquitetura do DSI-RTree não é necessário fixar o valor de M, pode-se encontrar o seu valor com base na quantidade de polígonos e no grau de preenchimento, de forma que o nó raiz do índice fique próximo de sua capacidade máxima. Pode-se reduzir a altura do índice para 2, 3 ou 4 níveis, reduzindo a troca de

mensagens e a latência do processamento das consultas, garantindo uma boa seletividade e paralelismo, desde o nível raiz do índice. Mesmo com no máximo quatro níveis é possível armazenar datasets de até 10 milhões de polígonos com M=100, aproximadamente.

Com base nestes parâmetros, a equação 1 é proposta para cálculo aproximado do valor de M, com base na quantidade de polígonos do dataset e em seu grau de preenchimento. Nela, a é a altura do índice desejada, n a quantidade de polígonos e p o percentual de preenchimento dos nós (entre 0 e 1).

$$M = \sqrt[a]{\frac{n}{p^{a-1}}} \qquad (1)$$

5. Avaliação de Desempenho

Testes foram realizados na implementação da arquitetura, a fim de avaliar a eficiência do índice distribuído em: i) Escalar horizontalmente à medida que mais estações de trabalho são adicionadas ao sistema; ii) Maximizar a quantidade de requisições atendidas por segundo e evitar processamento desnecessário (*overhead*); iii) Distribuir os dados uniformemente pelas máquinas para o aproveitamento do *hardware* disponível.

O ambiente de execução do *cluster* foi configurado da seguinte forma: i) 20 Máquinas Intel Core 2 Duo 2.4 GHz, com 1 Gb de memória RAM, utilizadas como Servidores DSI; ii) 20 Máquinas Intel Core 2 Duo 2.4 GHz, com 1 Gb de memória RAM, utilizadas como simuladores de Clientes do Serviço. Um simulador de cliente é uma máquina que executa requisições ao serviço, em uma determinada taxa por segundo, simulando o que ocorreria em um ambiente de produção; iii) 1 Máquina Intel Core 2 Duo 2.4 GHz, com 2 Gb de memória RAM, utilizada para comunicação entre os Servidores DSI; iv) Switch Gbit Dell PowerConnect 6248.

Em cada uma das máquinas foi instalada a distribuição Open Suse, versão 11.2, com Kernel Linux 2.6.31.12. A máquina virtual Java utilizada foi a Java(TM) SE Runtime Environment (build 1.6.0_20-b02). Nas máquinas simuladoras de clientes, os parâmetros -Xmx e -Xms (*Heap* e *Stack*) foram ajustados para o máximo possível (650Mb), considerando o uso parcial da memória pelo Sistema Operacional e a fragmentação da memória total disponível (1Gb) causada pelo processo de *boot*.

5.1. Qualidade do Índice Distribuído

O teste de Qualidade do Índice avalia dois aspectos do comportamento do sistema: i) A fração dos dados acessados (espaço de busca), em relação ao total de dados disponíveis, durante a execução de consultas; ii) Na fração acessada, qual o percentual de falsos positivos (dados acessados, que não serão retornados como resposta).

De modo geral, quanto maior for a redução no espaço de busca, melhor é o desempenho do sistema, pois menos acesso físico ao conjunto de dados é necessário. Tais valores foram coletados para todos os datasets e diferentes tamanhos de M. A média de cada um deles é apresentada na Figura 6. Os dados foram normalizados em percentuais, devido ao valor de M causar variação na quantidade de nós diretórios e folhas, e devido a essas mesmas quantidades serem diferentes de um dataset para outro. A escala do eixo y foi adaptada de 0 a 40% para melhor visualização. Os demais 60% correspondem à continuação da legenda de dados não acessados.

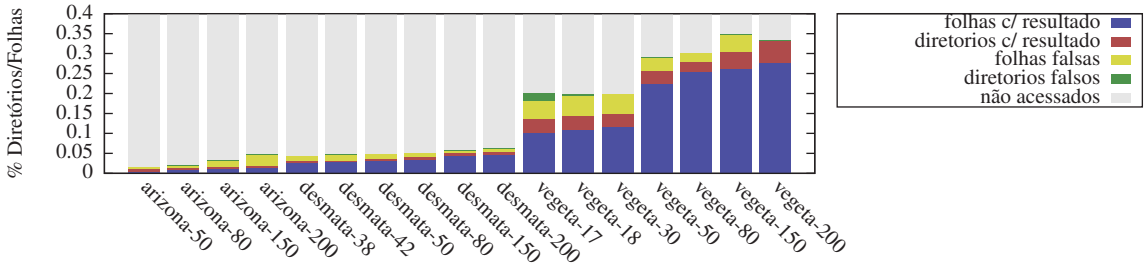

Figura 6. Percentual de Folhas e Diretório Acessados durante Buscas de Janela

Nota-se no gráfico que o acesso ao dataset está sempre abaixo de 5% para o dataset arizona e desmata, e abaixo de 35% para o dataset vegeta. O aumento na quantidade de acesso entre os datasets, principalmente no dataset vegeta está relacionado com a configuração das janelas utilizadas, que interceptam uma maior parte do dataset total. Como as janelas interceptam uma área maior e maior quantidade de itens do dataset, espera-se o acesso a um maior percentual de nós. Possivelmente, devido à maior interseção de espaço morto e sobreposição, o número de acessos poderia aumentar de forma desproporcional ao aumento da janela, mas isso não ocorreu.

Enquanto as janelas do dataset vegeta são maiores que as do dataset arizona 3,22 vezes, a quantidade de acesso aumenta praticamente na mesma proporção: 3,24 vezes (média para os quatro valores de M's). Entre o dataset vegeta e o dataset desmata a proporção é ainda menor: aumento da janela de 22,5 vezes contra aumento no acesso de 7,63 vezes. O índice distribuído apresentou uma ótima qualidade de acesso, pois consegue isolar e encontrar de forma eficiente o dado procurado que intercepta com janela de buscas grandes ou pequenas para datasets de diferentes tamanhos.

5.2. Avaliação do Distribuidor *Round-Robin*

A distribuição realizada pelo distribuidor tem impacto direto na quantidade de requisições atendidas e escalabilidade do sistema. Um distribuidor inadequado, pode concentrar todo o processamento em uma pequena fração das máquinas, fazendo com que surja um gargalo, mesmo existindo várias máquinas ociosas.

Para analisar este comportamento, mediu-se o acesso às máquinas por cada janela no dataset arizona. O resultado é apresentado nos mapas de calor da Figura 7. Um ponto preto no mapa (x, y) indica que determinada janela (x) não acessou nenhum nó na máquina (y). O nível de acesso varia conforme a legenda à direita de cada mapa. Observando a sequência de valores de uma linha completa – y – é possível verificar o acesso a uma máquina específica. Se uma máquina não for acessada por nenhuma busca de janela, existiria uma linha totalmente preta no mapa. Observando as colunas ao invés das linhas, pode-se identificar quais máquinas processaram cada uma das buscas de janela.

A distribuição das partições realizada pelo distribuidor *Round-Robin* explorou o processamento de todas as máquinas (nenhuma linha totalmente preta), não acumulando o processamento em poucas máquinas. Para o cenário de aplicações LBS isto provê escalabilidade ao sistema devido ao uso homogêneo do *cluster*, mesmo utilizando janelas de consulta seletivas. No entanto, em outros algoritmos de busca, o *Round-Robin* pode não ser adequado por causar excesso de comunicação.

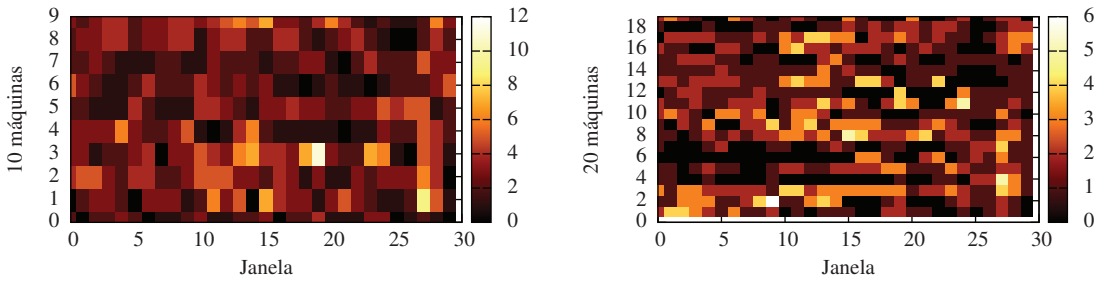

Figura 7. Mapa de calor do acesso às partições: dataset arizona, 10 e 20 máq.

5.3. Escalabilidade Horizontal

O Teste de Escalabilidade Horizontal é uma forma de medir o quão eficiente o sistema é em aproveitar novas máquinas a sua disposição, para aumentar sua capacidade ou vazão. Se a proporção entre o aumento de máquinas e a capacidade é de 1:1, diz-se que ela é linear, ou seja, dobrando-se a capacidade de processamento, dobra-se também a vazão de requisições atendidas. Uma proporção linear ou superior é desejável, mas nem sempre pode ser atingida devido à necessidade de sincronização de tarefas distribuídas no *cluster* e a limitação física de dispositivos como a rede de comunicação.

Foram utilizadas 5, 10, 15 e 20 máquinas para cada um dos datasets. Em cada combinação de datasets $versus$ quantidade de máquinas foi executado um conjunto de testes, com uma taxa fixa de requisições por segundo e início sincronizado entre todos os clientes. Cada requisição consiste no envio de uma janela pela aplicação cliente ao serviço. As janelas interceptam parte do dataset conforme ilustrado na Figura 4. O sistema procura no índice pelos polígonos que possuem interseção com a mesma (Busca de Janela), e os envia como resposta para a aplicação cliente. O resultado da execução dos testes é ilustrado no gráfico da Figura 8. As barras são agrupadas pela quantidade de máquinas utilizadas durante o teste e a altura de cada barra vertical representa a quantidade de requisições atendidas por segundo para cada dataset.

Observando o gráfico, há um aumento na quantidade de requisições por segundo atendidas pelo sistema, a cada adição de máquinas. Aumentando de cinco para dez máquinas, ou seja dobrando a capacidade de processamento, a escalabilidade é maior que linear para os três datasets. A escalabilidade continua maior que linear para os datasets arizona e desmata quando ocorre o aumento de 10 para 15 máquinas, com uma pequena redução em relação ao aumento anterior. Com o aumento de 15 para 20 máquinas, ape-

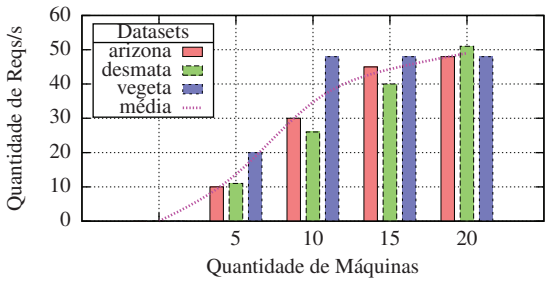

Figura 8. Escalabilidade Horizontal do DSI-RTree em 5, 10, 15 e 20 máq.

sar de ainda haver um ganho de escalabilidade, ele deixa de ser linear. Nota-se que a curva do fator de escalabilidade vai decrescendo à medida que se aumenta a quantidade de máquinas. Isto ocorre devido o custo de comunicação e em função da subutilização das máquinas que estão armazenando somente um pequeno subconjunto dos dados.

O dataset vegeta apresenta um fator de escalabilidade nulo a partir de 10 máquinas. Apesar das partições geradas para o dataset serem bem distribuídas, a pequena quantidade de polígonos é insuficiente para gerar um número significativo de partições para distribuir entre as máquinas, fazendo com que o custo de comunicação domine o tempo de resposta das consultas. Essa é uma indicação de que há um limite de divisão de datasets, o qual é atingido mais rapidamente em datasets pequenos.

5.4. Processamento de Grandes Datasets

Outra variável avaliada é a degradação da quantidade de requisições atendidas por segundo, à medida que datasets maiores são utilizados. A Escalabilidade Horizontal aliada à Qualidade do Índice, apresentada na Seção 5.1, deve permitir o processamento de datasets cada vez maiores, mantendo estável ou com pequena degradação a quantidade de requisições processadas por segundo. Em outras palavras, se a seletividade do índice é boa, processar buscas de janela também seletivas em datasets grandes ou pequenos requer um esforço semelhante, já que a redução no espaço de busca é eficiente.

Conforme ilustrado no gráfico da Figura 8, as três barras verticais que representam os datasets testados estão em níveis semelhantes em cada uma das quatro quantidades de máquinas. Em alguns casos (cinco e dez máquinas) a quantidade de requisições processadas para o dataset arizona é maior que o dataset desmata, apesar deste segundo possuir quantidade inferior de polígonos. A quantidade de requisições por segundo entre eles permanece praticamente igual em cada conjunto de máquinas testado, apesar do aumento do tamanho dos datasets: desmata maior que vegeta 15,2 vezes; arizona maior que vegeta 73,9 vezes e maior que desmata 4,8 vezes.

6. Trabalhos Correlatos

De forma geral, poucos aspectos da construção e processamento do índice têm sido investigados pelos trabalhos correlatos, de acordo com nosso conhecimento e observando as revisões de literatura feitas em [Manolopoulos et al. 2006, Jacox and Samet 2007]. Nosso trabalho apresenta novidades não discutidas na literatura como parâmetros importantes para definição do valor de M e suas implicações na escalabilidade e desempenho do sistema, aspectos do controle da comunicação através do correto dimensionamento da altura dos índices e o balanceamento do índice distribuído através do Distribuidor e a possibilidade de construção do índice de forma paralela com a consequente necessidade de bloqueios distribuídos para garantir a consistência dos índices gerados.

A primeira publicação encontrada a respeito do tema foi [Patel et al. 1997], a qual descreve a implementação de um ambiente completo de banco de dados paralelo para arquitetura NOW (*Network of Workstations*) chamado Paradise. A ênfase do trabalho é o processamento de imagens de satélites (*raster images*), ao contrário do foco deste trabalho que visa o processamento de dados vetoriais.

Na M-RTree [Koudas et al. 1996], a discussão é focada em datasets estáticos, cujos objetos espaciais são totalmente conhecidos previamente e o índice e suas partições

são montados de forma *bottom-up*. Outro problema observado é a grande concentração de processamento em um servidor mestre, que além de realizar as comparações de todos os nós diretórios durante as buscas, é utilizado como agregador de resposta para as aplicações clientes. Estes comportamentos também são evidenciados na MC-RTree [Schnitzer and Leutenegger 1999]].

Em [An et al. 1999], os autores exploram a distribuição de uma árvore R-Tree em um *cluster*. O artigo propõe uma arquitetura similar à M-RTree e MC-RTree e, por isso, possui os mesmos problemas descritos anteriormente. O amplo conjunto de experimentos realizados e a definição de uma taxonomia para a R-Tree distribuída foram as grandes contribuições dadas pelo artigo.

As duas propostas posteriores encontradas SD RTree e Hadoop-GIS [Mouza et al. 2007, Kerr 2009] abordaram aspectos muito diferentes dos apresentados anteriormente. Na SD-RTree uma árvore binária é utilizada ao invés de uma R-Tree, o que causa um grande aumento de mensagens. A proposta Hadoop-GIS apresenta uma alternativa para processamento paralelo de dados espaciais, mas não emprega índices para melhorar o desempenho das operações nos datasets.

7. Conclusão

Para que aplicações SIG e LBS possam ser, de fato, implantadas e utilizadas por muitos usuários é fundamental que os sistemas de geoprocessamento sejam escaláveis. A infra-estrutura de *cluster* de computadores com muitos nós já está disponível na *Computação nas Nuvens*, e o que ainda falta são infra-estruturas de *softwares* capazes de explorar o potencial de processamento e armazenamento para prover serviços de forma geral.

Outros projetos de geoprocessamento distribuído correlatos foram propostos na literatura [Koudas et al. 1996, Schnitzer and Leutenegger 1999, An et al. 1999, Mouza et al. 2007, Kerr 2009], mas esses, em sua maioria, não discutem o fator de escalabilidade horizontal de suas propostas, as implicações da definição do tamanho de partição dos dados no índice distribuído, inserção ou atualização de dados de forma concorrente no sistema e a qualidade do índice distribuído, conforme apresentado neste trabalho.

A principal contribuição deste trabalho consiste de uma plataforma capaz de escalar horizontalmente em número de clientes simultâneos e no tamanho dos datasets manipulados, o que foi demonstrado pelos testes realizados em ambiente real de um *cluster*. Outra contribuição deste trabalho é a implementação do núcleo do DSI-RTree, pois este, além dos resultados demonstrados, permitirá a experimentação de diferentes algoritmos de processamento de dados espaciais em larga escala, viabilizando a criação e avaliação de novos algoritmos de distribuição de dados espaciais, algoritmos de *join* distribuídos, processamento de operações espaciais de forma paralela, processamento de *workflow* com muitas operações espaciais encadeadas, dentre outros.

Apesar dos resultados apresentados neste trabalho confirmarem o potencial de escalabilidade, eles ficaram limitados pela capacidade de processamento das máquinas e da rede utilizados. Em testes preliminares realizados em um *cluster* de máquinas HPC nas *Nuvens da Amazon*, foram obtidos excelentes resultados, com consultas de janela para um dataset da Navteq de Pontos de Interesse (e.g., Restaurantes, Postos de Gasolina) da cidade de São Paulo, conseguindo atender até 1500 requisições por segundo com um tempo de resposta máximo de 1.81 segundo.

Referências

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

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

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

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

Bernhardsen, T. (2002). *Geographic information systems: an introduction*. Wiley.

Chen, Y. et al. (1997). A study of concurrent operations on R-trees. *Information Sciences*, 98(1-4):263–300.

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

DeWitt, D. and Gray, J. (1992). Parallel database systems: the future of high performance database systems. *Communications of the ACM*, 35(6):85–98.

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

Jacox, E. H. and Samet, H. (2007). Spatial join techniques. *ACM Trans. Database Syst.*, 32(1):7.

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

Kanth, R., Serena, D., and Singh, A. (2002). Improved concurrency control techniques for multi-dimensional index structures. In *Parallel Processing Symposium, 1998. IPPS/SPDP 1998.*, pages 580–586. IEEE.

Kerr, N. (2009). Alternative Approaches to Parallel GIS Processing. *Arizona State University - Master Thesis*.

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

Manolopoulos, Y., Nanopoulos, A., and Theodoridis, Y. (2006). *R-trees: Theory and Applications*. Springer Verlag.

Mouza, C., Litwin, W., and Rigaux, P. (2007). SD-Rtree: A Scalable Distributed Rtree. In *Proc. of IEEE ICDE Conference, Istanbul, Turkey*, pages 296–305.

Patel, J., Yu, J., Kabra, N., Tufte, K., Nag, B., Burger, J., Hall, N., Ramasamy, K., Lueder, R., Ellmann, C., et al. (1997). Building a scaleable geo-spatial DBMS: technology, implementation, and evaluation. In *Proceedings of the 1997 ACM SIGMOD international conference on Management of data*, page 347. ACM.

Schnitzer, B. and Leutenegger, S. (1999). Master-client R-trees: A new parallel R-tree architecture. In *ssdbm*, page 68. Published by the IEEE Computer Society.

Weiser, M. (1995). The computer for the 21st century. *Scientific American*, 272(3):78–89.