Um Estudo Sobre o Uso de Abordagens de Gerência de Dados em Sistemas de Análise Visual de Dados Espaço-Temporais*

Lorenna Christ'na Nascimento, Marcos Lage, Daniel de Oliveira

[1]Instituto de Computação – Universidade Federal Fluminense (UFF)

lorennachrist@id.uff.br,{mlage,danielcmo}@ic.uff.br

Abstract. *Over the last years, visual analytics systems have gained importance not only for presenting final results, but also for assisting in interactive analysis and decision-making processes. Such systems require efficient access to data, so that response times does not interfere with the user's ability to observe and analyze. Concurrently, research in the database domain has proposed solutions that can be used to support visualization systems. This paper presents a study of data management approaches to support interactive visualizations. We compared the performance of PostgreSQL, MonetDB, and Spark SQL to process multiple spatiotemporal queries common in visual analytics system. This study showed that Spark SQL presented the best performance for the chosen queries.*

Resumo. *Nos últimos anos, os sistemas de análise visual de dados têm ganho importância não só para apresentação de resultados, mas também para auxiliar em análises interativas e processos de tomada de decisão. Tais sistemas requerem que o acesso aos dados seja realizado no menor tempo possível para que não interfira na capacidade de observação e análise do usuário. Concorrentemente, pesquisas na área de banco de dados têm proposto soluções que podem ser usadas para apoiar sistemas de visualização. Esse artigo apresenta um estudo de abordagens de gerência de dados com o objetivo de apoiar visualizações interativas. Comparamos o desempenho do PostgreSQL, MonetDB e Spark SQL para processar múltiplas consultas espaço-temporais comuns em sistemas de análise visual de dados. O estudo mostrou que o Spark SQL apresentou o melhor desempenho para as consultas analisadas.*

1. Introdução

Com a popularização da área de Ciência de Dados (CD), o uso de técnicas de visualização tem ganho bastante importância [Schmidt 2020]. Apesar da área de CD envolver múltiplas disciplinas como Gerência de Dados e Aprendizado de Máquina (AM), a área de *Visualização* (daqui em diante referenciada somente como *Vis*) desempenha um papel fundamental, já que apoia a compreensão e análise dos dados de entrada, dos resultados de consultas e de modelos de AM. Em muitos casos, esse processo de exploração e análise visual requer o uso de *Vis* interativas [Munzner 2014], *i.e.*, capazes de reagir de forma "imediata" a uma determinada ação realizada no sistema (*e.g.*, um filtro temporal ou a seleção de uma área no espaço). Mais precisamente, para produzir *Vis* interativas, os dados devem ser acessados, filtrados e agregados de forma eficiente para que uma nova representação visual seja produzida com o mínimo possível de latência, idealmente abaixo de 0,5 segundos [Liu and Heer 2014].

Neste contexto, um dos desafios de construir *Vis* interativas é que filtros e agregações comumente envolvem consultas no espaço e no tempo. Os dados espaciais envolvidos nessas

*Os autores gostariam de agradecer a FAPERJ, CAPES e CNPq por financiarem o artigo.

operações consistem em coleções de pontos, linhas, regiões, retângulos, *etc.* [Samet 1990], e podem ser usados para representar estruturas mais complexas como bairros e cidades. Assim, muitos sistemas de *Vis* se utilizam de estruturas especializadas como os *Nano Cubes* [Lins et al. 2013] para representar dados espaço-temporais. Essas estruturas já se mostraram efetivas para reduzir a latência nas consultas, mas possuem algumas desvantagens como a indisponibilidade do dado no mais fino grão e o custo de armazenamento exponencial em relação ao número de atributos. Aliadas a tais estruturas, ainda podem ser utilizadas técnicas de processamento aproximado de consultas [Zimbrao and de Souza 1998].

Concorrentemente, a comunidade de Banco de Dados vem propondo uma série de soluções agnósticas de domínio ou problema para gerência de dados nos últimos anos. Tais soluções vão desde SGBDs NoSQL (*e.g.*, MonetDB [Boncz et al. 2005]) até *frameworks* de consulta e análise de dados de larga escala como o Spark SQL [Armbrust et al. 2015], além de SGBDRs como o PostgreSQL. Apesar não terem sido projetadas para apoiar *Vis* interativa, tais soluções têm se mostrado cada vez mais eficientes e podem ser usadas neste contexto, substituindo soluções como os *Nano Cubes*.

Esse artigo apresenta um estudo comparativo do desempenho de múltiplas soluções de gerência de dados para apoiar sistemas de *Vis* interativa. Escolhemos como estudo de caso uma ferramenta de análise de dados pluviométricos da cidade de Niterói denominada `TEMPO` [Nascimento et al. 2021]. A `TEMPO` permite ao usuário submeter consultas espaço-temporais com o objetivo de analisar o comportamento das chuvas em Niterói. O sistema possui dados de precipitação medidos em múltiplas estações pluviométricas ao longo de 13 anos (mais detalhes na Seção 4.1). Os resultados mostram que *frameworks* de processamento de consultas como o Spark SQL apresentaram o melhor desempenho.

2. *Vis* Interativa e Padrões de Consultas Espaço-Temporais

Sistemas de exploração e análise visual de dados têm sido usados com sucesso para estudar problemas mal-postos – problemas para os quais não é possível encontrar soluções utilizando técnicas puramente computacionais – ou para investigar temas que exigem o julgamento e/ou a expertise de um especialista de domínio [Munzner 2014]. Existem sistemas de *Vis* usados com sucesso em diversas áreas, como aplicações relacionadas a saúde [Caban and Gotz 2015] e gerenciamento urbano [Zheng et al. 2016]. Um dos requisitos fundamentais destes sistemas é que eles sejam capazes de atingir tempos de resposta interativos. [Liu and Heer 2014] mostram que latências superiores a 0,5 segundos já interferem na capacidade de observação, generalização e construção de hipóteses durante a exploração visual de dados.

Sendo assim, um dos requisitos comuns a todos sistemas de *Vis* Interativa é alcançar um tempo de resposta das consultas inferior a 0,5 segundos sempre que possível. O grande desafio é que tais consultas em sistemas de *Vis* comumente envolvem múltiplas junções, agregações e buscas no espaço e no tempo, o que faz com que as consultas se tornem complexas. [Doraiswamy and Freire 2020] descrevem uma série de consultas espaço-temporais comumente utilizadas em análises de dados espaço-temporais. Estas consultas são agrupadas em cinco classes: (C1) *Seleção*, (C2) *Junção*, (C3) *Agregação*, (C4) *Vizinho Mais Próximo* e (C5) *Consultas Geométricas*. A seguir detalhamos cada padrão no contexto do estudo de caso utilizado nesse artigo.

As consultas da classe de Seleção são aquelas em que uma série temporal é retornada de acordo com algum critério espacial definido, *e.g.*, retornar a série temporal de índices pluviométricos de uma estação pluviométrica específica. As consultas da classe de Junção são

aquelas em que junções são necessárias, sejam elas entre polígonos ou entre polígonos e pontos, *e.g.,* descobrir quais estações pluviométricas (pontos) se encontram em uma determinada área no mapa (polígono). As consultas da classe de Vizinho Mais Próximo retornam a partir de um ponto no espaço um outro ponto mais próximo de acordo com algum critério, *e.g*, dada uma coordenada geográfica, descobrir qual é a estação pluviométrica mais próxima. As consultas da classe de Agregação são aquelas em que funções de agregação são utilizadas, *e.g.*, descobrir a média de precipitação nos últimos dois meses de uma determinada estação. Finalmente, as Consultas Geométricas são aquelas que envolvem por exemplo a decomposição de um dado espaço determinado pela distância para uma determinada família de objetos (subconjuntos) no espaço, *e.g.,* usando o diagrama de Voronoi. É importante ressaltar que uma consulta complexa pode ser obtida pela composição das consultas anteriores.

3. Trabalhos Relacionados

Os *Nano Cubes* [Lins et al. 2013] são estruturas em memória projetadas para apoiar sistemas de *Vis* interativa. Os *Nano Cubes* podem ser usados para gerar visualizações bem conhecidas, como mapas de calor (*Heat Maps*), histogramas e gráficos de coordenadas paralelas. A vantagem dos *Nano Cubes* se comparados aos tradicionais *Data Cubes* [Battle et al. 2016] é que eles são capazes de armazenar até bilhões de pontos sem necessitar usar o disco, enquanto que os *Data Cubes* são capazes de gerenciar um conjunto de dados bem menor sem utilizar o disco. Entretanto, tais estruturas devem ser reconstruídas para cada coleção de atributos e tipos de agregação demandados pela aplicação, o que pode ser laborioso. Além disso, como os dados já se encontram pré-agregados em memória, qualquer consulta que necessite acessar o dado em seu grão mais fino não poderá ser realizada. Por fim, o tamanho da estrutura cresce exponencialmente com o número de atributos considerados.

Ao contrário de [Lins et al. 2013] e [Battle et al. 2016] que optaram por estruturas especializadas em memória para apoiar sistemas de *Vis*, [Jiang et al. 2018] propõem que tais sistemas utilizem SGBDRs como abordagem de gerência de dados. No artigo, os autores executam a análise de tempo de resposta das consultas espaço-temporais no SGBDR PostgreSQL. [Eichmann et al. 2020] propõem um *benchmark* chamado IDEBench para avaliar sistemas de *Vis* interativa. Os autores propõem uma série de consultas sintéticas e avaliam o benchmark com o SGBD MonetDB. Similarmente, [Battle et al. 2020] propõem um novo *benchmark* baseado em *traces* de múltiplos usuários que representem *workloads* com diferentes características de exploração de dados. Apesar de representar um avanço, o trabalho de [Battle et al. 2020] não explora o uso de *frameworks* de análise de dados como o Spark SQL.

4. Avaliação Experimental

4.1. Estudo de Caso

Utilizamos como estudo de caso a ferramenta `TEMPO` [Nascimento et al. 2021], cujo objetivo é captar dados pluviométricos multi-fonte com diferentes granularidades e integrá-los em um único repositório para posterior consulta e análise visual por parte de especialistas. A `TEMPO` se encontra no contexto do projeto "*Niterói Organizada e Segura: Estudo do Impacto das Chuvas*", realizado em parceria com a Prefeitura de Niterói no contexto do Programa de Desenvolvimento de Projetos Aplicados (PDPA). Além de armazenar os dados brutos (*e.g.*, em formato CSV, JSON, *etc*.), a `TEMPO` realiza as devidas transformações nos dados (*e.g.*, ajuste de granularidade, padronização de endereços e localizações, agregações, *etc*) e gera uma tabela de preparação única com os dados ajustados (em uma área de *staging*). A partir dessa tabela,

Tabela 1. Consultas executadas pela TEMPO

Consulta	Descrição	Classe	# Tuplas
Q1	Retornar a série temporal completa de precipitações em 11/2018 para todas as estações	C1, C2	193.604
Q2	Retornar a série temporal completa de precipitações em 2017, 2018 e 2019 para estações a uma distância de até 5km de um ponto específico	C2, C5	138.442
Q3	Retornar a série temporal completa de precipitações em 2017 para a estação mais próxima de um ponto específico até 5km	C2, C4	62.729
Q4	Retornar a soma das precipitações em 2018 para todas as estações	C1, C2, C3	392
Q5	Retornar a média das precipitações em 2016 para estações a uma distância de até 5km de um ponto específico	C2, C3, C5	83
Q6	Retornar a média das precipitações em 2016 para a estação mais próxima de um ponto específico	C2, C3, C4	3.389

os dados são carregados em um *Data Warehouse* (DW) [Kimball and Ross 2002], chamado de DW-TEMPO. Em sua versão original o DW-TEMPO foi implementado seguindo o modelo ROLAP (*Relational* OLAP), e utilizou-se o SGBDR PostgreSQL.

O *schema* do DW-TEMPO possui uma tabela fato e quatro tabelas de dimensão (*schema* estrela). A tabela fato *Chuvas* contém os valores de índices pluviométricos recebidos das estações (valores de interesse para análise), além de chaves estrangeiras para as dimensões. O DW-TEMPO possui as dimensões *Estação* (que define qual estação realizou a leitura do índice), *Localidade* (que define o local onde uma determinada estação se encontra), *Fonte* (que define qual foi a origem do dado carregado, *e.g.*, CEMADEN) e *Tempo* (que define a qual momento do tempo a leitura se refere). Por restrições de espaço não apresentamos o *schema* completo, mas informações adicionais sobre o DW-TEMPO podem ser obtidas em [Nascimento et al. 2021]. O *dump* do DW-TEMPO contendo os dados utilizados nesse artigo pode ser obtido em https://bit.ly/3dcnApY. Baseado no padrão de submissão de consultas ao DW-TEMPO na TEMPO, extraímos 6 consultas executadas pelo sistema durante as sessões de uso. É importante ressaltar que os dados no DW-TEMPO já se encontram pré-agregados na dimensão temporal. A Tabela 1 apresenta o código de cada consulta, sua descrição e sua classe, utilizando a definição de [Doraiswamy and Freire 2020].

4.2. Abordagens de Gerência de Dados Escolhidas e Ambiente de Execução

Os experimentos foram executados em três SGBDs/*Framework*: o PostgreSQL versão 13, o MonetDB versão v11.39.17 e o Spark SQL versão 3.11. A ideia foi explorar a heterogeneidade das soluções no que tange o modelo de dados adotado, o processamento de consultas, *etc*. O PostgreSQL é um tradicional SGBD de código aberto que segue o modelo relacional, enquanto que o MonetDB é um SGBD colunar que oferece suporte a SQL. Já o Apache Spark SQL é um componente desenvolvido no *stack* do Spark que disponibiliza a abstração chamada *DataFrame*, para apoiar a execução de consultas em SQL. Para os experimentos executados neste artigo, foi realizado o *deploy* de todas as soluções em um ambiente virtualizado com as seguintes características: Intel Core i5-7200U CPU @ 2.50GHz, 4GB de RAM, 80GB de Disco e Sistema Operacional Ubuntu 20.04.2 LTS. No caso do PostgreSQL não foi utilizado o PostGIS e todos os cachings estavam desabilitados.

4.3. Resultados

A Figura 1 apresenta a média do tempo de processamento de 20 execuções de cada consulta apresentada na Tabela 1 (a primeira execução de cada consulta foi descartada para efeitos de cálculo da média). Como podemos perceber pelas descrições apresentadas, as consultas Q2, Q3, Q5 e Q6 são da classe C4 ou C5. Em todas elas, se faz necessário o uso de funções e operadores matemáticos (*e.g.*, *radians* no PostgreSQL), que aumentam o tempo de execução das

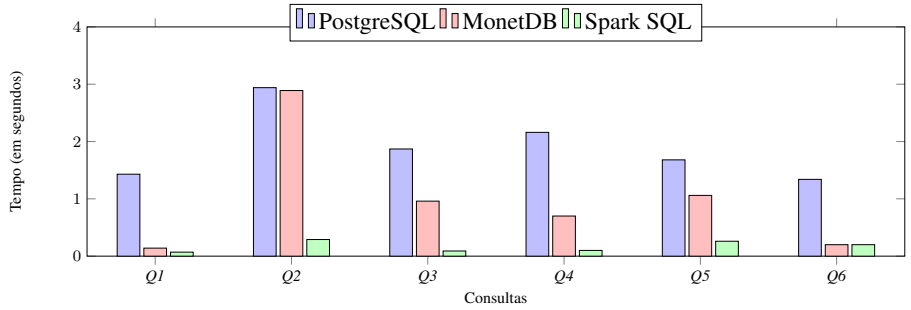

Figura 1. Tempo de Execução das Consultas Q1-Q6

consultas (em especial na consulta Q2 que retorna a série temporal completa além de realizar os cálculos de distância). Além disso, as consultas Q4, Q5 e Q6 fazem uso de funções de agregação (*e.g.*, *AVG*), o que também aumenta o tempo de execução das consultas. Podemos perceber que o PostgreSQL é o que apresenta o pior resultado em todas as consultas submetidas. Mesmo com a criação de índices nas chaves estrangeiras da tabela fato, e em atributos que favorecem consultas com igualdades (*e.g.*, "ano = 2017"), as consultas de classe C4, C5, ou as que retornavam um grande volume de dados, obtiveram tempos de execução maiores que os 0,5 segundos esperados. O MonetDB apresentou resultados ligeiramente melhores que o PostgreSQL em todas as consultas, mas ainda com um tempo superior a 0,5 segundos nas consultas Q2, Q3, Q4 e Q5. Além da questão do uso de funções e operadores matemáticos, em alguns casos em que múltiplos atributos são retornados, o MonetDB apresentou um tempo de execução da consulta maior. Já o Spark SQL apresentou tempos de execução menores que 0,5 segundos em todas as consultas. É importante mencionar que o `DW-TEMPO` possui aproximadamente 4,1 GB (a tabela fato contém 12.401.780 tuplas) e cabe quase que em sua totalidade em memória. Assim, o Spark consegue realizar praticamente todo o processamento em memória utilizando *Data Frames*, o que faz com que a execução se torne mais eficiente se comparada as outras abordagens que realizam acesso ao disco. Dessa forma, o Spark SQL se mostrou a abordagem recomendada para processar consultas na `TEMPO`. É importante ressaltar que esse resultado se justifica já que a `TEMPO` submete consultas tipicamente OLAP, e os dados não sofrem atualizações (uma vez que a `TEMPO` só carrega dados enviados por sensores). Como próximos passos, pretendemos avaliar outras abordagens como o Apache Drill e o BigDAWG (sistemas *Polystore*), além de avaliar conjuntos de dados maiores e processamento paralelo em ambientes distribuídos.

5. Conclusão e Trabalhos Futuros

Muitas aplicações da área de Ciência de Dados fazem uso de técnicas de *Vis*. Em muitos casos, essas aplicações requerem o uso de *Vis* interativas, que tradicionalmente definem um limite de 0,5 segundos como tempo máximo de resposta a uma ação na aplicação. Esse tempo de resposta deve englobar o acesso, filtragem e agregação dos dados. Apesar de existirem técnicas e estruturas de dados específicas para apoiar tais visualizações, as abordagens de gerência de dados propostas nos últimos anos podem ser usadas para apoiar sistemas de *Vis*. Nesse artigo realizamos um estudo comparativo de abordagens de gerência de dados com o objetivo de apoiar *Vis* interativas. Comparamos o desempenho de soluções conhecidas como o SGBDR PostgreSQL, o SGBD NoSQL MonetDB e *framework* Spark SQL para processar múltiplas consultas espaço-temporais da ferramenta `TEMPO` para visualização de dados pluviométricos. Das 6 consultas elencadas, o *framework* Spark SQL se mostrou o único capaz de processar todas as consultas em menos de 0,5 segundos. A grande vantagem do Spark SQL é que ele realiza o processamento em memória, se beneficiando da estrutura de *Data Frames* do Spark. O MonetDB também foi

capaz de responder a parte das consultas em menos de 0,5 segundos, mas apresentou uma perda de desempenho em consultas que envolviam funções e operadores matemáticos (*e.g.*, *radians*). Trabalhos futuros incluem a avaliação de novas abordagens, incluindo soluções *Polystore* como Apache Drill e o BigDAWG.

Referências

Armbrust, M., Xin, R. S., Lian, C., Huai, Y., Liu, D., Bradley, J. K., Meng, X., Kaftan, T., Franklin, M. J., Ghodsi, A., and Zaharia, M. (2015). Spark sql: Relational data processing in spark. In *SIGMOD*, page 1383–1394, New York, NY, USA.

Battle, L., Chang, R., and Stonebraker, M. (2016). Dynamic prefetching of data tiles for interactive visualization. In *SIGMOD*, pages 1363–1375. ACM.

Battle, L., Eichmann, P., Angelini, M., Catarci, T., Santucci, G., Zheng, Y., Binnig, C., Fekete, J.-D., and Moritz, D. (2020). Database benchmarking for supporting real-time interactive querying of large data. In *SIGMOD*, page 1571–1587, New York, NY, USA.

Boncz, P. A., Manegold, S., and Rittinger, J. (2005). Updating the pre/post plane in monetdb/xquery. In Florescu, D. and Pirahesh, H., editors, *XIME-P*.

Caban, J. J. and Gotz, D. (2015). Visual analytics in healthcare–opportunities and research challenges.

Doraiswamy, H. and Freire, J. (2020). A gpu-friendly geometric data model and algebra for spatial queries. In *SIGMOD*, page 1875–1885, New York, NY, USA.

Eichmann, P., Zgraggen, E., Binnig, C., and Kraska, T. (2020). Idebench: A benchmark for interactive data exploration. In *SIGMOD*, page 1555–1569, New York, NY, USA.

Jiang, L., Rahman, P., and Nandi, A. (2018). Evaluating interactive data systems: Workloads, metrics, and guidelines. In *SIGMOD*, page 1637–1644, New York, NY, USA.

Kimball, R. and Ross, M. (2002). *The Data Warehouse Toolkit: The complete guide to dimensional modeling*. Wiley, New York.

Lins, L. D., Klosowski, J. T., and Scheidegger, C. E. (2013). Nanocubes for real-time exploration of spatiotemporal datasets. *IEEE TVCG*, 19(12):2456–2465.

Liu, Z. and Heer, J. (2014). The effects of interactive latency on exploratory visual analysis. *IEEE transactions on visualization and computer graphics*, 20(12):2122–2131.

Munzner, T. (2014). *Visualization analysis and design*. CRC press.

Nascimento, L. C., Knust, L., Santos, R., Sá, B., Moreira, G., Freitas, F., Moura, N., Lage, M., and Oliveira, D. (2021). Análise de dados pluviométricos multi-fonte baseada em técnicas olap e de visualização: uma abordagem prática. In *WCAMA*, pages 1–10.

Samet, H. (1990). *The Design and Analysis of Spatial Data Structures*. Addison-Wesley.

Schmidt, J. (2020). Usage of visualization techniques in data science workflows. In *Proc. of the VISIGRAPP*, pages 309–316.

Zheng, Y., Wu, W., Chen, Y., Qu, H., and Ni, L. M. (2016). Visual analytics in urban computing: An overview. *IEEE Transactions on Big Data*, 2(3):276–296.

Zimbrao, G. and de Souza, J. M. (1998). A raster approximation for processing of spatial joins. In *VLDB*, pages 558–569. Morgan Kaufmann.