CellDB: an efficient database architecture for cellular spaces

Gilberto Ribeiro de Queiroz, Gilberto Câmara, Antônio Miguel Vieira Monteiro, Pedro Ribeiro de Andrade

[1]Divisão de Processamento de Imagens (DPI) – Instituto Nacional de Pesquisas Espaciais (INPE)
São José dos Campos – SP – Brazil

[2]Centro de Ciência do Sistema Terrestre (CCST) – Instituto Nacional de Pesquisas Espaciais (INPE)
São José dos Campos – SP – Brazil

[1]{gribeiro, gilberto, miguel}@dpi.inpe.br

[2]{pedro.andrade}@inpe.br

Abstract. *In recent years, the volume of data available to researchers performing computer simulations that represent spatially explicit dynamics has increased considerably. New technologies have made it possible to acquire data with increasingly fine resolutions and higher temporal frequency, which are stored in various formats. However, most of simulation tools have limited capacity for handling large volumes of data. By analyzing several models in the literature based on cellular representation of space, we identified common access patterns and data manipulation in these models. Taking this into consideration, this paper presents a flexible architecture for databases designed for modeling systems that deal with cellular representations. This architecture allows to work with large volumes of data, accessing different data sources and provides resources to work efficiently with the different identified patterns.*

Resumo. *Nos últimos anos, o volume de dados disponíveis para os pesquisadores realizarem simulações computacionais que representem dinâmicas espacialmente explícitas vem aumentando consideravelmente. Novas tecnologias têm possibilitado a aquisição de dados com resoluções cada vez mais finas e com maior frequência temporal, que são armazenados nos mais diversos formatos. No entanto, grande parte das ferramentas de simulação possuem limitada capacidade de manipulação de grandes volumes de dados. A partir do estudo de vários modelos consagrados da literatura que usam uma representação celular do espaço, foram identificados padrões de acesso e manipulação de dados comuns nesses modelos. Levando isso em consideração, este trabalho apresenta uma arquitetura flexível de bancos de dados dirigida a sistemas de modelagem que lidam com representações celulares. Esta arquitetura possibilita trabalhar com grandes volumes de dados, acessar diferentes fontes de dados, além de fornecer recursos para trabalhar de forma eficiente com os diferentes padrões identificados.*

1. Introdução

Nos últimos anos, o volume de dados disponíveis para os pesquisadores realizarem simulações computacionais representando dinâmicas espacialmente explícitas vem aumentando consideravelmente. Novas tecnologias têm possibilitado a aquisição de dados com resoluções cada vez mais finas e com maior frequência temporal, que são armazenados nos mais diversos formatos. Com isso, a comunidade científica encontra-se diante de vários desafios que vão desde o armazenamento e acesso a essas enormes bibliotecas digitais até o uso efetivo dos dados em análises e processamentos [1].

Em relação às ferramentas de simulação, grande parte possui limitada capacidade de manipulação de grandes volumes de dados. Uma possível forma de solução deste problema seria a integração entre as ferramentas de simulação e tecnologias especialmente projetadas para um gerenciamento eficiente de grandes volumes de dados. No entanto, os sistemas gerenciadores de bancos de dados relacionais ou SGBD-R [2], que representam a classe de tecnologia mais difundida ao longo dos últimos 40 anos, nunca se mostraram a solução ideal para esta integração [3]. Em consequência disso, torna-se relevante o projeto de novas tecnologias de armazenamento que possam suprir as lacunas desta comunidade.

A partir do estudo de vários modelos consagrados da literatura que usam uma representação celular do espaço, foram identificados padrões de acesso e manipulação de dados comuns nesses modelos. Levando isso em consideração, este trabalho apresenta uma arquitetura flexível de bancos de dados dirigida a sistemas de modelagem que lidam com representações celulares. Esta arquitetura possibilita trabalhar com grandes volumes de dados, acessar diferentes fontes de dados, além de fornecer recursos para trabalhar de forma eficiente com os diferentes padrões identificados. Adicionalmente, propomos um *benchmark* baseado em modelos computacionais consagrados na literatura para avaliar o desempenho e as limitações das tecnologias de bancos de dados para este fim.

2. Revisão da Literatura de Bancos de Dados

Nos últimos anos, a busca por sistemas de bancos de dados mais eficientes e especializados em certos nichos de aplicações tem estimulado o desenvolvimento de novas tecnologias com características diferentes dos sistemas relacionais (SGBD-R). Sistemas como o *Apache CouchDB* [4], *MongoDB* [5], *SciDB* [6], *Apache Cassandra* [7] e *Neo4J* [8] fornecem modelos de dados, linguagens de consulta e interfaces de programação específicos. A concepção desses sistemas tem levado em consideração desde o início uma forte preocupação com aspectos de escalabilidade em grande número de máquinas e alta disponibilidade [9]. Em consequência disso, usam técnicas de replicação e particionamento de dados não convencionais. Muitos não oferecem os mesmos recursos existentes nos SGBD-R, como operadores para realização de junções, suporte a transações *ACID* [2] e, principalmente, suporte à linguagem *SQL* [10]. Esses novos sistemas estão sendo denominados de **NoSQL**, acrônimo de *Not Only SQL*, que embora não seja um termo de total consenso, tem sido usado para enfatizar a diferença com os sistemas relacionais.

Atualmente existem centenas de sistemas NoSQL com as mais diversas formas de armazenamento, recursos e arquiteturas [11, 12]. A comunidade tem usado uma taxonomia para os sistemas de acordo com a forma de armazenamento:

(a) bancos de dados orientados a documentos (*Document Stores*);

(b) sistemas de armazenamento de pares chave-valor (*Key-Value Stores*);

(c) bancos de dados baseados em famílias de colunas (*Column Stores*);

(d) bancos de dados baseados em grafos (*Graph Databases*);

(e) bancos de dados com modelos e linguagens matriciais (*Array Databases*).

Os sistemas da categoria *a*, em geral, utilizam a notação JSON (*JavaScript Object Notation*) para representação dos dados. *JSON* é um formato leve para intercâmbio de dados baseada na linguagem *JavaScript* que popularizou-se entre os desenvolvedores de aplicações Web. Os documentos não possuem obrigatoriamente um esquema global compartilhado o que possibilita definir dinamicamente novos atributos sem a necessidade de alteração de outros documentos previamente armazenados. Atualmente, os dois sistemas baseados em software livre mais populares pertencentes a essa classe de NoSQL são o *MongoDB* e o *Apache CouchDB*.

Conhecidos por *Key-Value Stores*, os sistemas da categoria *b* tem origem nas ideias introduzidas pelo *DBM* [13]. Tratam-se de bibliotecas com rotinas para o gerenciamento de bancos de dados baseadas no armazenamento de pares chave-valor, onde tanto a chave quanto os valores podem ser qualquer cadeia de *bytes*, sendo de responsabilidade das aplicações a definição de funções de comparação entre as chaves e o correto tratamento dos valores. Quase todos os sistemas são fortemente baseados em árvores-b^+ [14] e *hash* [2]. Atualmente, as implementações mais conhecidas são: *Oracle Berkeley DB* [15], *Kyoto Cabinet* [16] e *LevelDB* [17].

Os sistemas de bancos de dados baseados na Teoria dos Grafos [18], da categoria *c*, fornecem três construtores básicos em seu modelo de dados: nós (ou vértices), ligações (ou arestas) e propriedades. Os nós são usados para modelar objetos que existem de forma independente das ligações. Em geral não possuem um esquema rígido. Assim dois objetos representados por nós de um mesmo grafo podem ter atributos bem distintos. As arestas são usadas para materializar o relacionamento entre nós. As propriedades podem ser usadas tanto para descrever atributos de nós quanto de arestas. A maior partes destes sistemas de bancos de dados têm optado por integrar uma linguagem para travessia de grafos denominada de *Gremlin* [19] ao invés do uso estrito da SQL. As implementações mais conhecidas dos *Graph Databases* são: *OrientDB* e o *Neo4J* [20].

Na categoria *d*, sistemas como o *Apache Cassandra* [21] oferecem um modelo de programação mais sofisticado que os *key-value stores*. Além da noção de chaves existem os grupos/famílias de colunas e super-famílias de colunas.

Os *Array Databases* (categoria *e*) representam uma tecnologia relativamente nova de bancos de dados, desenvolvida em resposta às demandas das várias comunidades científicas. O *SciDB* [22, 23] representa o projeto de pesquisa e implementação mais promissor nesta área.

Apesar da grande quantidade de sistemas de bancos de dados disponíveis, existe um *gap* entre as ferramentas de simulação e as tecnologias de bancos de dados. Em grande parte, isto se deve ao fato de que as primitivas necessárias a um armazenamento e recuperação eficientes de dados para ferramentas de simulação não seguem o mesmo

padrão de acesso das aplicações convencionais. É preciso considerar características particulares como computação com grande iteratividade sobre os dados dos modelos e manipulação de grandes volumes de dados. No caso dos modelos espaciais, ainda é relevante a representação de estruturas de relacionamento espacial, dificilmente implementada de forma eficiente em SGBD-R.

3. Modelos Computacionais

Quatro modelos dinâmicos espaciais são usados para definir os requisitos de software que serão usados na elaboração da arquitetura do CellDB. Estes modelos são uma amostra representativa dos modelos dinâmicos espaciais existentes na literatura.

O primeiro modelo é o jogo da vida, ou *game of life*, originalmente proposto por *John Conway* [24]. Este modelo tem como o objetivo simular o processo de nascimento e morte de organismos em uma sociedade. Cada célula de um espaço matricial possui um estado vivo ou morto, que indica a presença ou ausência de um indivíduo. A simulação se inicia com uma determinada configuração espacial de células vivas e computa a evolução desta população com o passar do tempo. Cada célula atualiza o seu estado usando regras determinísticas baseadas nos estados dos seus oito vizinhos:

- Uma célula viva morre de solidão se possuir menos de dois vizinhos vivos.
- Uma célula viva morre por superpopulação se possuir mais de três vizinhos vivos.
- Uma célula morta se transforma em viva se possuir exatamente três vizinhos vivos.
- Uma célula viva mantem o seu estado se possuir com dois ou três vizinhos vivos.

Note que este processo ocorre em paralelo no tempo para cada célula. Assim, a alteração de estado de uma determinada célula não pode afetar o estado das células vizinhas em um mesmo instante de tempo.

O segundo modelo, proposto por *Thomas Schelling*, estuda processos de segregação espacial [25]. Neste modelo, células brancas, vermelhas e pretas são distribuídas sobre um espaço celular. As vermelhas e pretas representam indivíduos nas suas moradias, enquanto que as brancas são áreas desocupadas. Qualquer indivíduo tolera viver em uma vizinhança que possua indivíduos da cor oposta, mas apenas até o limite de metade da sua vizinhança. Caso algum indivíduo esteja em minoria na sua vizinhança, ele decide se mudar para uma nova célula na qual ele esteja em maioria ou igualdade. A cada instante de tempo, um indivíduo verifica a sua satisfação e decide se mudará ou não. Este processo se repete iterativamente até que todos os membros da população estejam satisfeitos.

O terceiro modelo investiga o processo de desmatamento na Amazônia brasileira. Este modelo é uma versão simplificada do modelo proposto por [26]. Diferentes propriedades do espaço são usadas para computar o potencial de uma célula ser desmatada, como por exemplo distância para a estrada mais próxima, porcentagem de área protegida e a média de desmatamento das células vizinhas. O modelo então aloca

uma quantidade pré-determinada de desmatamento de acordo com os potenciais de cada célula.

O quarto e último modelo descreve um processo hidrológico a partir de um espaço celular fictício [27]. Cada célula possui um atributo estático que representa a altitude da célula e um atributo dinâmico que indica a quantidade de água na mesma. O modelo simula o processo de escorrimento de água nesta região a partir de uma chuva nas células mais altas. Usando a proximidade espacial entre as células e as suas altitudes, um fluxo de água é simulado para cada célula, que envia água para as suas vizinhas de menor altitude.

Analisando estes modelos, pode-se observar que existem estruturas comuns que podem ser gargalos quando é necessário manipular grandes volumes de dados. Todos os modelos usam uma estrutura de vizinhança para computar o estado futuro de cada célula. Esta estrutura cresce proporcionalmente mais do que o número de células quando se aumenta a resolução do espaço celular. O jogo da vida e o modelo hidrológico processam os dados em paralelo para cada célula. Desta forma, é necessário um passo de sincronização do modelo no qual todos os estados e valores são atualizados simultaneamente. Este passo demanda considerável memória uma vez que ele necessita gerar cópias de um ou mais atributos para todo o espaço celular.

4. CellDB

O *CellDB* é uma arquitetura flexível de bancos de dados voltada para ferramentas de simulação computacional que usam representações de espaços celulares. O objetivo é possibilitar a criação de uma nova geração de aplicativos capazes de realizar a simulação de modelos que operam sobre grandes volumes de dados espaciais, considerando a capacidade de máquinas *desktop* convencionais. Os elementos desta arquitetura são focados nas operações de E/S e alocação de memória RAM durante a simulação de modelos.

A idéia central da arquitetura mostrada na Figura 1 é fornecer uma camada de abstração de dados para as ferramentas de simulação (*Modeling Toolbox*). Esta arquitetura possui uma API com construtores básicos necessários à representação e manipulação de espaços celulares e seus elementos. Os principais conceitos desta API são:

(a) *cell*: representa uma célula com seus atributos do estado passado e presente;

(b) *cell_space*: estrutura de agregação que contém um conjunto de células e suas relações de vizinhança;

(c) *neighboorhod*: materialização dos relacionamentos de vizinhança entre células e seus atributos;

(d) *cell_iterator*: abstração para realização de travessias sobre o conjunto de células de um espaço celular;

(e) *input_data_source*: fonte de dados de origem do espaço celular, podendo compreender SGBD-R, arquivos de imagem (*raster*), bancos de dados no formato interno do CellDB, entre outros;

(e) *output_data_source*: fonte de dados de destino, usada para armazenamento final dos dados após o término da simulação de um modelo;

(f) *cell_batch_insert*: possibilita a implementação eficiente de estratégias de construção de espaços celulares;

(g) *cell_batch_update*: possibilita a implementação eficiente de estratégias de extração de atributos a serem associados às células de um espaço celular a partir de operações espaciais;

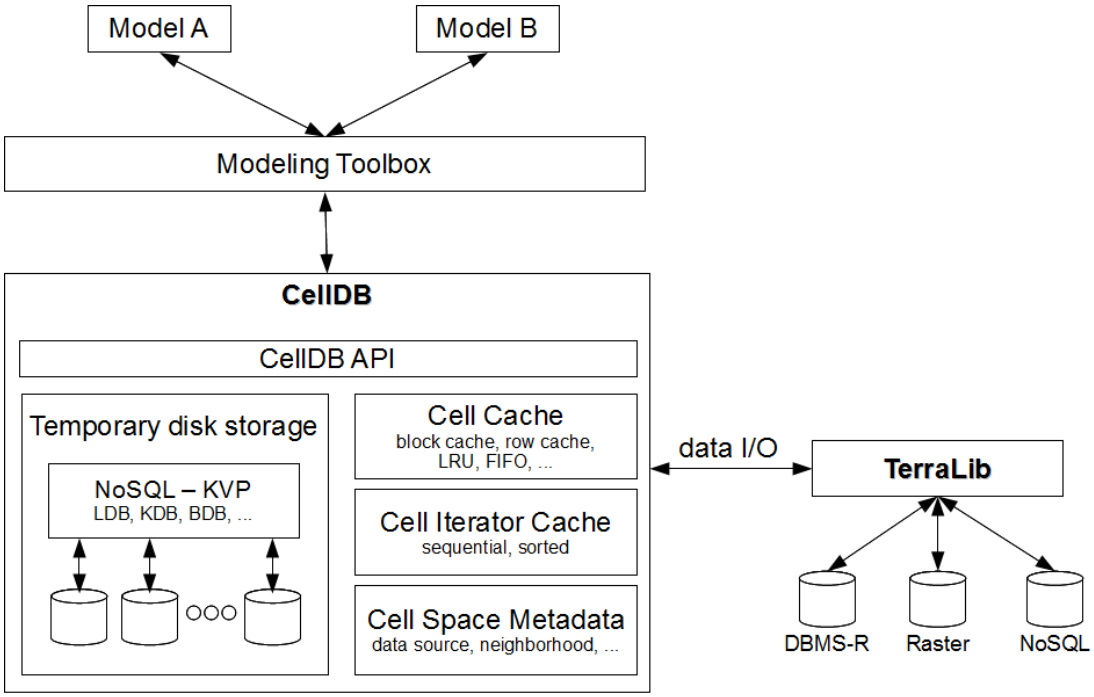

Figura 1 - Macro arquitetura CellDB

Para cada espaço celular são mantidos alguns metadados básicos: o repositório de origem do espaço celular; o sistema de referência espacial; retângulo envolvente dos objetos do espaço celular; a resolução espacial das células; as dimensões do espaço celular; informações específicas do *driver* de acesso a dados. No nível interno, existem dois caches de dados distintos que podem utilizar diferentes políticas de substituição de objetos (FIFO, LRU, *Slide Window*, *Neighbor Block*). A política usada irá depender do tipo de modelo e pode ser um parâmetro informado pela ferramenta de simulação. Os dois níveis de cache possibilitam responder consultas de acesso aleatório às células assim como realizar travessias eficientes sobre todas as células de um determinado espaço celular.

A arquitetura permite armazenar temporariamente as células durantes a execução dos modelos. Este módulo apoia-se em sistemas NoSQL da classe dos *key-value stores*. Com estes sistemas é possível evitar gargalos de comunicação cliente/servidor; realizar acessos aleatórios eficientes à células, uma vez que eles utilizam árvores-b^+ ou funções *hash* para indexar os dados; percorrer todas as células de forma eficiente. Como cada tipo de modelo requer diferentes estratégias de representação e acesso a dados, existe uma implementação específica para cada um deles. Entre as alternativas incluem-se: (a)

armazenamento apenas do valor presente das células; (b) armazenamento dos valores presente e passado, com leituras dos atributos passado e escrita no presente; (c) vizinhanças armazenadas; (d) armazenamento por linhas; (e) armazenamento por blocos de células;

5. Considerações Finais

A arquitetura proposta na seção 4 foi implementada em C++ no ambiente da TerraLib [27]. Os modelos descritos na seção 3 estão sendo usados para avaliar a arquitetura com diferentes sistemas de bancos de dados, estratégias de armazenamento e tamanhos de espaços celulares. Os resultados preliminares têm indicado que a arquitetura é capaz de lidar com grandes volumes de dados de forma eficiente.

References

[1] BERRIMAN, G. B.; GROOM, S. L. How Will Astronomy Archives Survive the Data Tsunami? **Communications of the ACM**, v. 54, n. 12, p. 52-56, December 2011.

[2] ELMASRI, R.; NAVATHE, S. B. **Fundamentals of database systems**. Addison Wesley, 2006.

[3] GRAY, J.; LIU, D. T.; NIETO-SANTISTEBAN, M.; SZALAY, A.; DEWITT, D. J.; HEBER, G. Scientific Data Management in the Coming Decade. **ACM SIGMOD**, v. 34, n. 4, December 2005.

[4] ANDERSON, J. C.; LEHNARDT, J.; SLATER, N. **CouchDB: The Definitive Guide**. O'REILLY, 2010. 272p.

[5] CHODOROW, K.; DIROLF, M. **MongoDB: The Definitive Guide**. O'REILLY, 2010. 216p.

[6] BROWN, P. G. Overview of SciDB: large scale array storage, processing and analysis. In: ACM SIGMOD International Conference on Management of data, 2010. **Proceedings...** ACM, New York, NY, USA, 2010. p. 963-968.

[7] LAKSHMAN, A.; MALIK, P. Cassandra: a decentralized structured storage system. **ACM SIGOPS Operating System Review**, v. 44, n. 2, p. 35-40, April 2010.

[8] NEUBAUER, P. **Neo4J Spatial - GIS for the rest of us**. Disponível em: <http://www.slideshare.net/peterneubauer/2011-07oscon>. Acesso: 08 maio 2012.

[9] CATTELL, R. Scalable SQL and NoSQL data stores. **ACM SIGMOD**, v. 39, n. 4, p. 12-27, December 2010.

[10] MELTON, J. **Advanced SQL: 1999 - Understanding Object-Relational and Other Advanced Features**. Morgan Kaufmann, 2003. 563p.

[11] WIKIPEDIA. **NoSQL**. Disponível em: <http://en.wikipedia.org/wiki/NoSQL>. Acesso: 03 Junho 2012.

[12] **List of NoSQL databases**. Disponível em: <http://nosql-database.org>. Acesso: 03 Junho de 2012.

[13] GNU. **GDBM – GNU dbm**. Disponível em: <http://www.gnu.org/software/gdbm>. Acesso: 14 agosto 2011.

[14] COMER, D. Ubiquitous B-Tree. **ACM Computing Surveys**, v. 11, n. 2, p. 121-137, June 1979.

[15] OLSON, M. A.; BOSTIC, K.; SELTZER, M. Berkeley DB. In: Annual conference on USENIX (ATEC '99). **Proceedings...** USENIX Association, Berkeley, CA, USA, 1999.

[16] DEAN, J.; GHEMAWAT, S. **LevelDB**. Disponível em: <http://code.google.com/p/leveldb>. Acesso: 08 agosto 2012.

[17] FAL LABS. **Kyoto Cabinet: a straightforward implementation of DBM**. Disponível em: <http://fallabs.com/kyotocabinet>. Acesso: 15 agosto 2011.

[18] BOLLOBÁS, B. **Modern graph theory**. New York, EUA: Springer-Verlag, 1998. 408p.

[19] AVRAM, A. Gremlin, a Language for Working with Graphs. **Info Q**, Jan, 2010. Disponível em: <http://www.infoq.com/news/2010/01/Gremlin>. Acesso: 01 junho 2012.

[20] HUNGER, M. Neo4j: Java-based NoSQL Graph Database. **Info Q**, Feb., 2010. Disponível em: <http://www.infoq.com/news/2010/02/neo4j-10>. Acesso: 09 junho 2012.

[21] Lakshman, A.; Malik, P. Cassandra: a decentralized structured storage system. **ACM SIGOPS Operating System Review**, v. 44, n. 2, p. 35-40, April 2010.

[22] CUDRE-MAUROUX, P.; KIMURA, H.; LIM, K.-T.; ROGERS, J.; SIMAKOV, R.; SOROUSH, E.; VELIKHOV, P.; WANG, D. L.; BALAZINSKA, M.; BECLA, J.; DEWITT, D.; HEATH, B.; MAIER, D.; MADDEN, S.; PATEL, J.; STONEBRAKER, M.; ZDONIK, S. A demonstration of scidb: a science-oriented dbms. **Proc. VLDB Endow**., v. 2, n. 2, p. 1534-1537, August, 2009.

[23] BROWN, P. G. Overview of sciDB: large scale array storage, processing and analysis. In: ACM SIGMOD International Conference on Management of data, 2010. **Proceedings...** ACM, New York, NY, USA, 2010. p. 963-968.

[24] GARDNER, M. Mathematical Games - The fantastic combinations of John Conway's new solitaire game "life". **Scientific American** 223:120–123, 1970.

[25] Schelling, T. C. Dynamic models of segregation. **Journal of mathematical sociology**, Taylor & Francis 1(2):143-186, 1971.

[26] Aguiar, A. P. D., 2006. Modeling Land Use Change in the Brazilian Amazon: Exploring Intra-regional Heterogeneity. Doctorade Thesis, INPE.

[27] CÂMARA, G.; VINHAS, L.; QUEIROZ, G. R.; FERREIRA, K. R.; MONTEIRO, A. M.; CARVALHO, M.; CASANOVA, M. TerraLib: An open-source GIS library for large-scale environmental and sócio-economic applications. In: HALL, B.; LEAHY, M (Eds). **Open Source Approaches in Spatial Data Handling**: Advances in Geographic Information Science. Springer, 2010. p. 247-270.