Accelerating Spatial Range Queries

Alexandros Stougiannis[¶], Farhan Tauheed[†‡], Thomas Heinis[†], Anastasia Ailamaki[†]

[¶]*Department of Computer and Systems Sciences, Stockholm University, Sweden*

[†]*Data-Intensive Applications and Systems Lab, École Polytechnique Fédérale de Lausanne, Switzerland*

[‡]*Brain Mind Institute, École Polytechnique Fédérale de Lausanne, Switzerland*

ABSTRACT

It is increasingly common for domain scientists to use computational tools to build and simulate spatial models of the phenomena they are studying. The spatial models they build are more and more detailed as well as dense and are consequently difficult to manage with today's tools. A crucial problem when analyzing spatial models of increasing detail is the scalable execution of range queries. State-of-the-art approaches like the R-Tree perform suboptimally on today's models and do not scale for more dense, future models. The problem is that the amount of overlap in the tree structure increases as a function of the level of detail/density in the model.

In this demonstration we showcase ZOOM, a new tool to efficiently execute spatial range queries on increasingly detailed (denser) models. ZOOM is based on FLAT, a novel range query execution approach that effectively decouples the query execution time from the density of the dataset, thereby ensuring efficient query execution. At the core of the demonstration thus is the visualization of the novel query execution strategy of FLAT which we contrast with a visualization of the query execution of the R-Tree.

1. INTRODUCTION

In many scientific disciplines it has become standard practice for scientists to simulate the phenomena they study on a supercomputer. The cycle of modeling the phenomena, simulating it and comparing the simulation results with the actual phenomena helps the scientists to better understand the phenomena. More and more precise instruments, a better understanding of the phenomena and the abundance of computational power lead scientists to build and simulate increasingly detailed/dense and big spatial models. Accessing, building and analyzing progressively dense models, however, challenges current data management tools. The ability to access the important parts of their data, i.e., their models, efficiently so they can analyze, build and simulate new models is crucial.

EDBT/ICDT '13 March 18 - 22 2013, Genoa, Italy

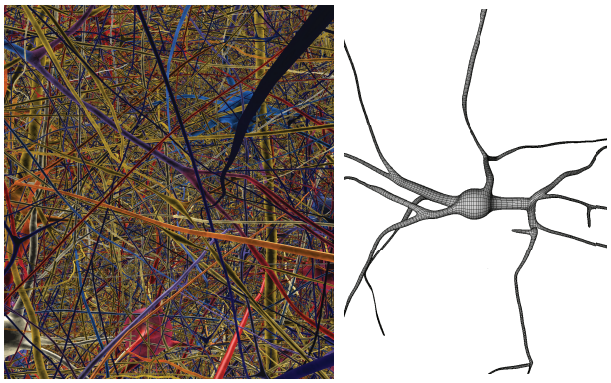

Figure 1: A visualization of a very small model featuring thousands of neurons (left) and a single neuron where the shape/morphology is modelled with a fine-grained surface mesh (right).

The neuroscientists we collaborate with in the Blue Brain Project (BBP [11]) suffer from the problem of dense models in their work. Based on data acquired in years of anatomical research, the neuroscientists in the BBP build and simulate biophysically realistic models of the rodent neocortex, the most detailed computer models of the brain to date. A visualization of a model is shown in Figure 1 (left). Today's models feature up to one million neurons, each modeled with its exact shape (represented with a very fine-grained surface mesh featuring millions of polygons - an example is shown in Figure 1 (right)) and electrophysiological properties.

Because the spatial models used are so fine-grained, each neuron is modeled with millions of mesh polygons, state-of-the-art methods do not support neuroscientists in the efficient analysis of the spatial brain models. The problem will only become worse as they model the brain increasingly fine-grained, modeling parts of it on the molecular level (e.g., the synapses). Similar trends, building ever finer-grained models, can also be seen in other disciplines, e.g., increasingly fine grained earthquake simulations [6, 9] or detailed models for simulating protein synthesis [13].

Motivated by the challenges the neuroscientists face we have developed FLAT, a novel spatial index. FLAT ensures that, although the models become more detailed and the datasets more dense, range queries can still be executed efficiently. This is not only important in neuroscience, but also in other scientific disciplines. The first experimental system

of FLAT was initially presented at ICDE 2012 [15] showing that it scales better than R-Tree based approaches on increasingly dense datasets from neuroscience as well as on datasets from other domains.

Contributions And Innovation: We demonstrate and visualize ZOOM, a tool to aid neuroscientists navigate through the brain. ZOOM is based on the novel execution strategy of spatial range queries of FLAT and illustrates its benefits in terms of faster query execution. FLAT in essence takes the ideas and principles of the B+-Tree and applies them to three dimensional data. Instead of adding pointers to the leaf nodes for the efficient execution of a range query in one dimension, FLAT uses the pointers in three dimensions. With this basic idea and through the careful design of the data structures, FLAT significantly outperforms several R-Tree approaches. Audience members are able to define queries and to see them executed on FLAT and the R-Tree, visualizing both strategies as well as monitoring the execution with statistics.

Visual Experience: The audience is able to interact with ZOOM through a graphical user interface to execute queries on a neuroscience dataset, i.e., the neocortex of a rat. Predefined queries can be executed but the audience can also define their own queries on a small visualization of the neocortex. Each query is executed on both FLAT as well as the R-Tree and during the simultaneous execution, statistics for both approaches are shown in real-time. More importantly, the query execution is explained while the query executes, impressively visualizing FLAT's novel execution strategy for spatial range queries and contrasting it with a visualization of the R-Tree execution strategy. The result of the query, a subset of the neocortex surface mesh, is also visualized.

2. RELATED WORK

Database research has produced many approaches to index volumetric spatial objects [4] in recent years.

Some of the approaches exploit particularities of the datasets indexed, e.g., for connected tetrahedral meshes [12].

Arguably the seminal data structure to index generic spatial objects is the R-Tree [7]. The R-Tree is a disk-based, multi-dimensional generalization of the B-Tree [2] that recursively encloses objects in minimum bounding rectangles (MBRs). Several extensions of the basic approach have been proposed, all, however, suffer to varying degrees from the problem of overlap and dead space. A first class addresses the overlap issue through repeated insertion [3] or through replication [14]. The second class, the bulk-loading R-Trees, assumes that all data is known a priori and exploits this to minimize the MBRs and thus to reduce overlap. Multiple bulk-loading approaches have been developed, some for datasets where the objects have rather uniform distribution of objects and objects sizes [8, 10] and others which work particularly well on extreme data [1, 5] (with respect to aspect ratio or distribution).

Despite the numerous improvements and approaches to alleviate the problem of overlap in the R-Tree, it still introduces considerable I/O overhead, thereby slowing down query execution. The denser a dataset is, i.e., the more spatial elements are in the same unit of space, the more overlap the bounding boxes of the R-Tree have, the bigger the I/O overhead becomes and consequently the slower query execution becomes.

3. FLAT QUERY EXECUTION

ZOOM uses at its core FLAT's query execution strategy to evaluate spatial range queries. The novel aspect of FLAT's execution strategy is its use of neighborhood information, i.e., what spatial elements neighbor each other, to speed up query execution independent of data density. In the following we briefly discuss how the neighborhood information is computed, how this information is stored and how it is used to execute range queries.

3.1 Indexing

To index a dataset, FLAT first groups spatially close objects on the same disk page. We refer to these disk pages as objects pages. For each object page FLAT computes the page MBR, the minimum bounding rectangle containing all objects on the page, and more importantly, the neighborhood information, i.e., pointers to other spatially close object pages. More precisely, if object page O_1 contains an element which is close to another element on page O_2, then FLAT stores a pointer from O_1 to O_2.

The page MBR and neighborhood information is stored in the seed index, a traditional spatial index (an R-tree): each page MBR is indexed with the seed index. In the leafs of the seed index, along with the page MBR we also store a reference to the object page as well as the neighborhood information, i.e, pointers to leaf nodes that contain spatially close page MBRs.

Figure 2 shows the data structures and their relationships. The rectangles at the bottom represent the object pages which pack spatially close objects while the arrows express neighborhood (i.e., an arrow between two object pages means the pages neighbor each other). The tree structure at the top represents the seed index and stores in its leaf nodes the MBRs of each object page. Furthermore, along with the MBR of each object page O it stores a pointer to the actual object page O as well as pointers to all neighbors of O. The pointers to the neighbors point to the leaf nodes where the neighbor information is stored in (i.e., MBR, pointer to object page etc.).

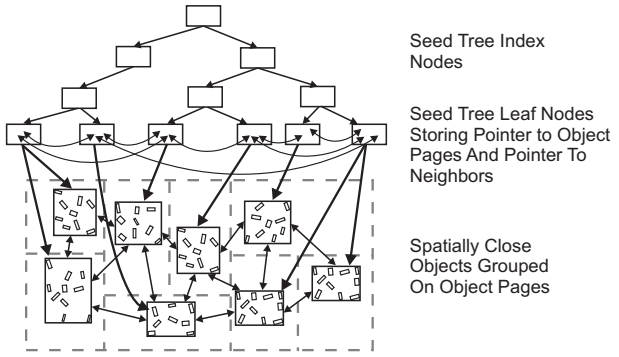

Figure 2: FLAT groups spatially close elements on object pages (rectangles at the bottom) and inserts the MBRs into the seed tree (tree structure at the top).

3.2 Querying

With the seed index containing page MBRs and neighborhood information, FLAT computes the result of range

queries in two phases:

- **Seed Phase:** In the first phase of the query execution, FLAT retrieves from the seed index an arbitrary page MBR that intersects with the query range. Retrieving an arbitrary element (page MBR) is a cheap operation, even from an R-tree which is subject to the problem of overlap in face of dense data. As opposed to retrieving a specific element or a range (and thus following many paths in the tree due to overlap), only one single path has to be followed from the root of the tree to one of the leafs to retrieve an *arbitrary* element in the query range. The complexity of this operation is hence typically in the order of the height of the R-Tree.
- **Crawl Phase:** Once an element in the query range has been retrieved, FLAT recursively follows the neighborhood links until no more page MBR can be found that intersects with the query. All payload pages referenced by the page MBRs are retrieved and all spatial objects on it are tested if they are in the query region.

Because both phases are independent of dataset density, the seed phase depends on the height of the seed index (R-Tree) and the crawl phase depends on the size of the result, the overall approach is independent of the data density and thus scales to more dense datasets.

4. DEMONSTRATION

The demonstration has two parts. In a first part we explain ZOOM and the approach of FLAT by way of a poster. In the second part the user can execute queries of which the result is visualized. More importantly, as the query is executed, the statistics of the query execution are calculated as well as shown and the functioning of the algorithms is visually illustrated.

4.1 Part I: Explanation of FLAT

In this part of the demonstration, we use a poster to explain the audience ZOOM and how FLAT at its core is able to decouple the execution time from the density of the spatial model/dataset. We explain in detail the limitations of tree-based approaches and explain FLAT's novel query execution strategy, i.e., the two phases of query execution, the seed phase which is independent of dataset density and the crawl phase which only depends on the size of the result set.

4.2 Part II: Executing Queries

ZOOM (and with it FLAT) is currently used in the workflow of the neuroscientists of the BBP in order to build models and to visualize the model. For the first, a plethora of small range queries is executed to test if any other element intersects, for the latter larger range queries are executed to retrieve whatever is in the field of vision of the user/neuroscientists.

In this demonstration users execute queries on both indexes simultaneously to emulate a race between the two indexes. We use real neuroscience data, i.e., a small part of the neocortex of the rat brain containing just 10'000 neurons.

4.2.1 *Interactive Query Execution*

In the second part of the demo the user can execute queries and the execution as well as the result is visualized. The queries are executed on both, FLAT and the R-Tree. The user can choose on what variant of the R-Tree (i.e, Hilbert [8], STR [10], Priority R-Tree [1] or TGS [5]) the query is executed.

Prepared Queries: To simplify the query execution for the user we have a selection of predefined queries that can be chosen. The predefined queries retrieve interesting areas of the rat's neocortex.

User Defined/Interactive Query: With ZOOM the user can also interactively define and execute his own queries in areas of interest. On a small visualization of the brain model the user can move around and resize a query box. Once content with the query, the user can choose to execute it. The query selection is shown in Figure 3.

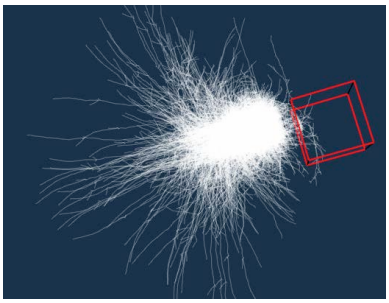

Figure 3: ZOOM's interactive query selection on a small part of the rodent neocortex.

4.2.2 *Visual Experience*

The visual experience of the demo has three components.

Result Visualization: The dataset indexed is a mesh representation of the 10'000 neuron model of the rat's neocortex. To illustrate the result and how it is retrieved by FLAT and the R-Tree, ZOOM continuously visualizes it with a three dimensional representation, i.e., every part of the result successively retrieved from disk through the index is added to the visualized model as the query runs.

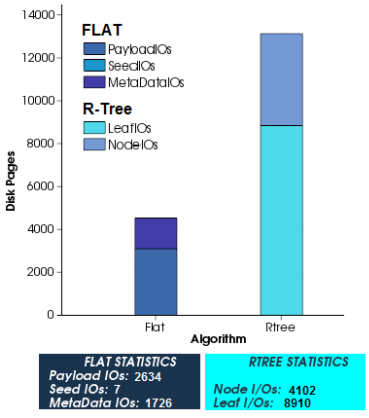

Figure 4: Statistics for both, R-Tree and FLAT, which are updated live during query execution.

Statistics: While executing the query, bar charts are continuously updated to show how many pages each approach retrieves from disk. For FLAT bar charts with the number of objects pages and seed tree disk pages are shown. For the R-Tree we show the number of leaf pages/nodes and internal pages/nodes retrieved, thereby illustrating the overlap in the R-Tree. As the screenshot in Figure 4 shows, detailed statistics including the time for the execution of both queries are shown.

Algorithm Illustration: The exact procedure of how the two algorithms work is shown in the demo. For FLAT we show how it recursively crawls the neighborhood of the seed element by depicting the bounding box of every disk page (the bounding box containing all elements on the page) loaded as is shown in Figure 5. With this the audience can clearly see how only the minimum disk pages needed are loaded from disk. In case of the R-Tree we also show the bounding box of every page/node retrieved from disk and color it so it becomes clear that some pages are retrieved multiple times due to overlap. To further illustrate the issue of overlap in the R-Tree variant chosen, we show in a histogram how many nodes are accessed on each level of the tree. The histogram clearly shows that nodes on the upper level (other than the leaf level) are accessed much more frequently than those on the leaf level, hence clearly showing overlap in the tree structure.

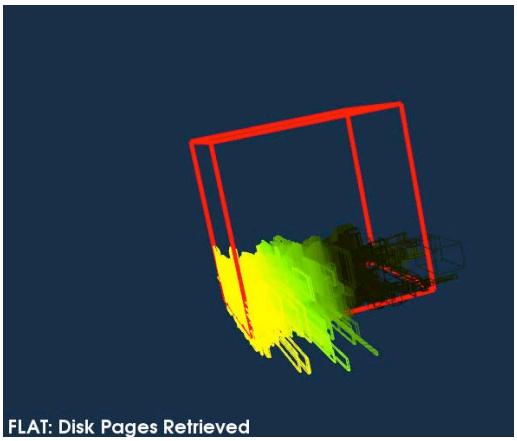

Figure 5: Illustration of the query execution strategy of FLAT, showing the order of the parts of the query result loaded, i.e., crawling through the result.

5. CONCLUSIONS

More precise instruments and the abundance of computational power combined with an ever better understanding of the phenomena leads scientists in general and neuroscientists in particular to build increasingly detailed spatial models of the phenomena under scrutiny. Doing so leads to an ever better understanding of the phenomena but, at the same time, also leads to massive amounts of dense spatial data that is difficult to analyze. As a consequence we have developed ZOOM, a tool to visualize neuroscience data, and FLAT, a spatial index with a novel range query execution strategy at its core. FLAT's query execution strategy decouples the time for executing queries from the density of the spatial model.

This demo showcases ZOOM, a tool for visualizing neuroscience data based on FLAT. Queries on real neuroscience are executed on FLAT and the R-Tree simultaneously while visualizing both execution strategies as the queries run. To demonstrate the benefits of FLAT, statistics (execution time, I/O etc.) are shown at query runtime as well. ZOOM also provides the user with a interactive graphical interface where queries can be defined. Similar to ZOOM's use by neuroscientists, the result of the query, the neurons in the query range, are visualized.

6. REFERENCES

[1] L. Arge, M. de Berg, H. J. Haverkort, and K. Yi. The Priority R-tree: a practically efficient and worst-case optimal R-tree. In *SIGMOD '04*.

[2] R. Bayer. Binary B-Trees for Virtual Memory. In *Proceedings of the Workshop on Data Description, Access and Control*, 1971.

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

[4] V. Gaede and O. Günther. Multidimensional Access Methods. *ACM Computing Surveys*, 30(2), 1998.

[5] Y. J. García, M. A. López, and S. T. Leutenegger. A Greedy Algorithm for Bulk Loading R-trees. In *GIS '96*.

[6] J. Gray, A. Szalay, A. Thakar, P. Kunszt, C. Stoughton, D. Slutz, and J. Vandenberg. Data Mining the SDSS SkyServer Database. In *Technical Report, MSR-TR-2002-01, Microsoft Research*, 2002.

[7] A. Guttman. R-Trees: A Dynamic Index Structure for Spatial Searching. In *SIGMOD '84*.

[8] I. Kamel and C. Faloutsos. Hilbert R-Tree: An Improved R-Tree using Fractals. In *VLDB '94*.

[9] D. Komatitsch, S. Tsuboi, C. Ji, and J. Tromp. A 14.6 Billion Degrees of Freedom, 5 Teraflops, 2.5 Terabyte Earthquake Simulation on the Earth Simulator. In *SC '03*.

[10] S. Leutenegger, M. Lopez, and J. Edgington. STR: a Simple and Efficient Algorithm for R-Tree Packing. In *ICDE '97*.

[11] H. Markram. The Blue Brain Project. *Nature Reviews Neuroscience*, 7(2):153–160, 2006.

[12] S. Papadomanolakis, A. Ailamaki, J. C. Lopez, T. Tu, D. R. O'Hallaron, and G. Heber. Efficient Query Processing on Unstructured Tetrahedral Meshes. In *SIGMOD '06*.

[13] K. Y. Sanbonmatsu, S. Joseph, and C.-S. Tung. Simulating movement of tRNA into the ribosome during decoding. *Proceedings of the National Academy of Sciences*, 102(44):15854–15859, 2005.

[14] T. K. Sellis, N. Roussopoulos, and C. Faloutsos. The R+-Tree: A Dynamic Index for Multi-Dimensional Objects. In *VLDB '87*.

[15] F. Tauheed, L. Biveinis, T. Heinis, F. Schürmann, H. Markram, and A. Ailamaki. Accelerating range queries for brain simulations. In *ICDE '12*.