Data-driven Neuroscience: Enabling Breakthroughs Via Innovative Data Management

Alexandros Stougiannis[¶] Farhan Tauheed[†‡] Mirjana Pavlovic[†]
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

Scientists in all disciplines increasingly rely on simulations to develop a better understanding of the subject they are studying. For example the neuroscientists we collaborate with in the Blue Brain project have started to simulate the brain on a supercomputer. The level of detail of their models is unprecedented as they model details on the subcellular level (e.g., the neurotransmitter). This level of detail, however, also leads to a true data deluge and the neuroscientists have only few tools to efficiently analyze the data.

This demonstration showcases three innovative spatial management techniques that have substantial impact on computational neuroscience and other disciplines in that they allow to build, analyze and simulate bigger and more detailed models. More particularly, we demonstrate a tool that integrates three spatial data management techniques that have enabled breakthroughs in neuroscience: FLAT that enables efficient querying of spatial data, SCOUT that allows for fast exploration of spatial data and TOUCH that makes efficient data discovery possible.

Categories and Subject Descriptors

H.2.8 [**DATABASE MANAGEMENT**]: Spatial databases and GIS; G.2.2 [**DISCRETE MATHEMATICS**]: Graph Theory

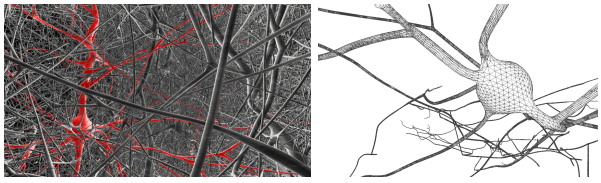
Keywords

Spatial Indexing; Spatial Join; Range Queries, Graph Prefetching, FLAT, TOUCH, SCOUT

1. INTRODUCTION

Scientists in many disciplines no longer solely study natural phenomena in vitro or in vivo, but instead attempt to understand the phenomenon better by simulating it on supercomputers or cloud deployments. Processing and analyzing the simulation data, i.e., the models simulated or the simulation output is a challenging task with today's tools. Efficient tools to build and analyze models in order to prepare new experiments is crucial for them. While the spatial indexes developed in the past [2] are of great help to scientists, several problems have not yet been addressed.

The work presented in this demo is motivated by our collaboration with the Blue Brain Project (BBP [10]). The neuroscientists in the BBP build biophysically realistic models of the rat brain at an unprecedented level of detail with data acquired in several years of anatomical research. A small model contains several thousands of neurons (a small part is shown in Figure 1, left), each one of which modelled with electrophysiological properties as well as a precise morphology defining the branches that extend into large parts of the tissue in order to receive and send out information to other neurons (Figure 1 (right) shows a cell morphology). The neuroscientists use a supercomputer (BlueGene/P with 16K cores) to simulate the propagation of electrical impulses through the branches of the neurons.



(a) Model (b) Neuron

Figure 1: Visualization of a small part of the model (left) and the surface mesh of a neuron (right).

In their quest to better understand the brain, i.e., how we perceive, feel etc., the neuroscientists need to build increasingly big and detailed models of it. Building models at the required level of detail and of the size needed, however, is impossible with today's tools. What the neuroscientists therefore urgently need are spatial data management techniques that (1) allow to efficiently query massive spatial brain models, (2) provide fast data exploration capabilities to analyze spatial data and (3) enable efficient discovery of synapses in their models.

Until recently, the models contained only up to 10'000 neurons but thanks to the data management tools we have developed (FLAT, SCOUT and TOUCH) models of one million neurons or bigger can be built and simulated today. One million neurons, however, is still far from their ultimate goal of building models as big as the human brain (~10^{11} neu-

SIGMOD'13, June 22–27, 2013, New York, New York, USA.

rons), but by addressing their data management challenges with the data management tools we demonstrate, we can help them to build ever bigger models. Although our motivation to build the tools comes primarily from our work with neuroscientists, our tools are nevertheless very useful for scientists in many other disciplines (biology [3], astronomy [4], geology [6], etc.) as well.

In the remainder of the paper we briefly present the spatial indexing techniques that are part of the demo, how we visualize them and how the audience can interact with them. We structure the paper in three main sections, one for each part of the tool.

2. EFFICIENT SPATIAL DATA QUERYING

A recurring type of query that scientists need to execute on their models are spatial range queries. The neuroscientists in the BBP use spatial range queries to build, analyze and visualize their models. Many indexes have been developed in the past to execute spatial range queries [2] and while they execute range queries efficiently on many spatial data sets, they fail to do so on dense datasets. The problem is that the denser the dataset is, i.e., the more elements are in the same unit of space, the more overlap and dead space tree-based indexes (R-Trees and variants: STR, TGS, PR-Tree, R+-Tree, R*-Tree and others) have. Several improvements have been proposed, each, however, has drawbacks, e.g., the R+-Tree [14] replicates elements to avoid overlap but thereby also increases the index size considerably.

2.1 FLAT: Efficient Range Query Execution

To address the problem of dense spatial datasets and hence the problem of increasingly detailed models we have developed FLAT [15], a new query execution strategy for dense datasets. At the core of FLAT is its novel range query execution strategy that uses neighborhood information, i.e., what spatial elements neighbor each other, to make the time for query execution independent of data density.

More precisely, to make FLAT work on arbitrary datasets it adds in the indexing phase information to the dataset describing what spatial elements neighbor each other. In the query phase, FLAT first uses an R-Tree (STR [9] bulk-loaded) to find an arbitrary element e inside the query range and then uses the neighborhood information to recursively visit the elements in the range: beginning with e, it recursively visits e's neighbors. The neighbors of retrieved elements that are not in the range are not visited.

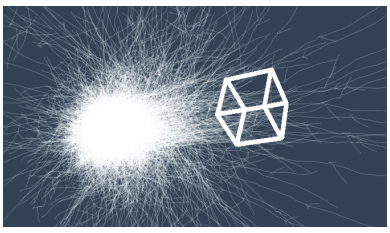


Figure 2: FLAT's interactive query selection on a small part of the rodent neocortex.

Both phases of the query execution are independent of the dataset density. Finding an *arbitrary* element in a query range typically only depends on the height of the R-Tree (as opposed to finding a particular element in an R-Tree which suffers from overlap [15]). Retrieving all neighboring elements in the query range by using the neighborhood information only depends on the size of the result. FLAT hence executes range queries more efficiently on today's dense datasets and will scale substantially better to future, even more detailed models.

FLAT is currently used by the neuroscientists to compute statistics (tissue density etc.) of the models they build and to visualize smaller parts of the model.

2.2 Interactive Visualization

The demo of FLAT compares the range query execution of FLAT and the R-Tree on real neuroscience data representing a small part of the rat neocortex (represented by a surface mesh). The audience can visually define a query in any region of a small representation of the neocortex (shown in Figure 2) and can test how FLAT and the R-Tree behave when executing queries in dense and sparse regions.

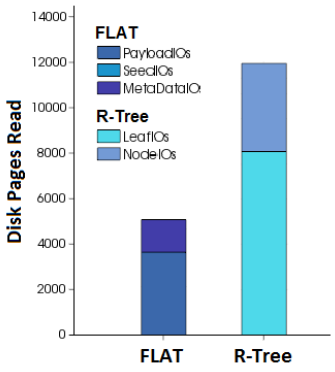


Figure 3: Statistics for both, R-Tree and FLAT, which are updated live during query execution.

Once the query is chosen, it is executed on both, FLAT and the R-Tree. While the query is running, the result is visualized as it is retrieved (similar to how the neuroscientists use it in practice) and also statistics (disk pages retrieved, time etc.) are shown and updated at runtime (shown in Figure 3).

At the same time we visualize the query execution strategy of FLAT. We show the order of elements retrieved in the query range and thereby illustrate how FLAT crawls through the query range by means of the neighborhood information (shown in Figure 4). We contrast this with the query execution strategy of the R-Tree and show for the R-Tree how many nodes are retrieved on each level (due to overlap more nodes are retrieved on higher levels).

3. EFFICIENT DATA EXPLORATION

In order to analyze spatial models of road networks, arterial trees etc. scientists need to execute moving range queries, i.e., they need to execute spatial range queries interactively in close succession on a model to analyze & visualize the neighborhood of the structure they are following. Neuroscientists need this type of query to follow neuron branches to see where they intersect other branches: at every step they retrieve the surroundings of the branch at a particular point and visualize it. Because this is an interactive process - data is retrieved, visualized and then analyzed by a user before the next query is executed - data can be prefetched to considerably speed up the query sequence.

State-of-the-art approaches do not prefetch spatial data with good enough accuracy because they rely on limited information. They either only use the current location [13] or the last few positions to predict the next query location. Because the structures of neurons (but also of arterial trees, lung airways, etc.) are irregular and jagged, current prediction approaches do not perform well. Other approaches learn from past user behavior to predict future positions [8]. For massive models like in our scenario, however, learning from past user behavior does not significantly improve pre-

diction accuracy because the probability that several users follow the same paths is small.

3.1 SCOUT: Content-Aware Prefetching

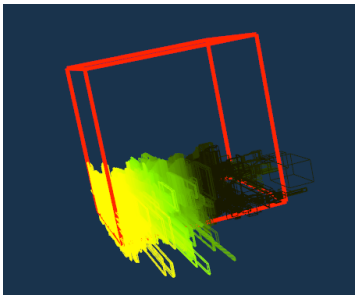


Figure 4: Illustration of FLAT's query execution strategy, showing the order of the parts of the query result loaded (by coloring them), i.e., crawling through the result.

As a consequence we have developed SCOUT [16], a prefetching method for spatial data that prefetches with a considerably higher accuracy speeding up query sequences by a factor of up to 15×.

Only considering the previous query locations is not enough to predict future query locations with high accuracy. To support scientists in executing spatial range query sequences along their path of interest, SCOUT thus not only considers the locations of previous queries, but also their content. While the result of query q in the sequence is loaded, SCOUT already starts to reconstruct the dominating structures/the topological skeleton in q and approximates them with a graph. Once the graph is constructed, it is traversed to find the locations where its edges exit q. At the exit locations, the edges exiting are extrapolated linearly to predict the next query locations. Range queries are then executed at the predicted locations to prefetch data into memory.

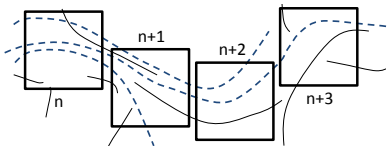


Figure 5: Pruning the irrelevant structures (solid lines) from the candidate set (dashed lines) in subsequent queries (solid squares) of the sequence.

In every query result there are many different structures and the scientist may be following any one of them. To identify the structure the scientist follows, SCOUT exploits that all queries in the spatial range query sequence must contain the structure followed. It thus only considers the intersection between the structures leaving the $(n-1)^{th}$ query and the set of structures entering the n^{th} (the most recent) query. The structure the user follows must be in the intersection. With several queries in a sequence, the structure the user follows can thus be identified reliably. Figure 5 shows how the candidate set is reduced and the structure followed is identified reliably to prefetch the correct data.

3.2 Prefetching Visualization

To demonstrate the benefits of SCOUT the audience can interactively walk through a model representing a small part of the rat brain. The dataset used represents several thousand neurons with surface meshes. Audience members can choose what prefetching method they want to use and can interactively walk through the model. With this they can test SCOUT and see that when following a structure, SCOUT prefetches nicely and the visualization is smoother compared to moving through the model randomly. Users can also compare SCOUT against other prefetching approaches (Hilbert prefetching [13], Extrapolation prefetching) and will see that due to the accurate prefetching the visualization is quicker and smoother than for other approaches.

Figure 6: SCOUT speeds up walk-through following a neuron branch through a model.

As the screenshot in Figure 6 illustrates, we also show statistics of the last visualization, i.e., how much data was prefetched in total, how much was correctly prefetched and how much data needed to be retrieved additionally. The statistics are refreshed as data is being prefetched to make the difference to other approaches more explicit. Additionally we visualize what parts of the model are prefetched between queries for the different prefetching methods.

4. EFFICIENT DATA DISCOVERY

When building a small model of the brain, the neuroscientists first put together several thousand neuron structures (an example of just one is illustrated in Figure 1 (right)) and then identify where to place the synapses, i.e., the places where branches of different neurons are close enough for electrical impulses to leap over [7]. The latter, placing the synapses, is a difficult data management problem best described as a distance join [5] on an unindexed and unsorted dataset to find pairs of neuron branches within distance e of each other.

To perform the spatial join at scale, the neuroscientists run it in the main memory of either a supercomputer (BlueGene/P) or a cluster. Only two approaches [11, 1] have been developed particularly for use in memory so far. Neither is particularly efficient: the nested loop join [11] has a complexity of $O(n^2)$ while the sweep line approach [1] can become inefficient if too many elements are on the sweep line (likely in case of dense data/detailed models). Approaches developed for disk [5] can of course also be used in memory, most of them, however, first need to index the dataset in a costly step before the spatial join can be performed or suffer from excessive memory use (due to object replication, e.g., PBSM).

4.1 TOUCH: In-Memory Spatial Join

Given the lack of scalable approaches, we have developed TOUCH [12], a new two-way spatial join approach that

works efficiently in memory and is one order of magnitude faster than known approaches (PBSM) and two orders of magnitude faster than known approaches with an equally small memory footprint (synchronized R-tree traversal, S3 and Scalable Sweep Join). TOUCH, our approach is designed radically different than known approaches in that it avoids space-oriented partitioning and thus also avoids element replication. Replication has to be avoided because it (a) increases the memory footprint and (b) requires multiple comparisons (as well as making the removal of duplicate results necessary).

Instead, TOUCH uses data-oriented partitioning to join datasets A and B. It first indexes dataset A and packs all elements into partitions to open up empty space between partitions, i.e., where there are no elements. These empty spaces allow TOUCH to use *filtering*, a concept used to reduce the number of comparisons: if any element b from dataset B falls into empty spaces, it can safely be ignored as there are by definition no elements from A in the empty space with which b could intersect. In the second phase TOUCH directly assigns objects of B to the data-oriented index of the A and by doing so avoids the problems caused by excessive index-overlap in other data-oriented approaches. The combination of data-oriented partitioning during index-building on the first data set with hierarchical assignment of the second dataset leads to significantly fewer comparisons and speeds up the spatial join.

4.2 Spatial Join Visualization

We visualize how TOUCH works on a small neuroscience dataset (that fits into main memory) and show how it finds the locations where synapses need to be placed faster. The demonstration of TOUCH has three elements. First the audience member chooses on a small representation of the brain model the area to perform the spatial join (same interface as Figure 2) as well as a method to perform the spatial join with, i.e., TOUCH, S3, PBSM etc.

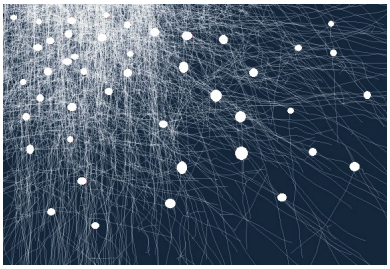


Figure 7: As the spatial join identifies locations of synapses, they are highlighted in a three dimensional model of the brain.

The execution of the spatial join is then visualized by highlighting on a three dimensional model of the brain the locations where synapses need to be placed as the join proceeds. Figure 7 shows how the locations of synapses are highlighted as the spatial join runs. Attendees can execute the join with different methods and can see how TOUCH outperforms the other approaches. To quantify and to make the difference more explicit we update at runtime charts showing time spent on the join, memory footprint as well as the number of pairwise comparisons needed.

5. CONCLUSIONS

The neuroscientists in the Blue Brain Project build increasingly big, complex and detailed models of the brain. But while the unprecedented size and level of detail of their models provide fundamentally new insight into neuroscience phenomena, the amounts of spatial data involved are also unprecedented and are difficult to handle with today's tools. The lack of efficient tools makes it difficult to analyze the data and therefore to build, analyze and fix the models.

With FLAT, SCOUT and TOUCH, this demo showcases spatial data management techniques we have developed to address some of the biggest data management challenges faced in the BBP. The techniques we demonstrate have enabled considerable breakthroughs in the BBP.

6. REFERENCES

[1] H. Edelsbrunner. *Algorithms in combinatorial geometry*. Springer-Verlag, 1987.

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

[3] S. Gnanakaran, H. Nymeyer, J. Portman, K. Y. Sanbonmatsu, and A. E. Garcia. Peptide folding simulations. *Current Opinion in Structural Biology*, 13(2):168–174, 2003.

[4] 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.

[5] E. H. Jacox and H. Samet. Spatial Join Techniques. *ACM TODS*, 32(1):7, 2007.

[6] 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 *Supercomputing '03*.

[7] J. Kozloski, K. Sfyrakis, S. Hill, F. Schürmann, C. Peck, and H. Markram. Identifying, Tabulating, and Analyzing Contacts Between Branched Neuron Morphologies. *IBM Journal of Research and Development*, 52(1/2):43–55, 2008.

[8] D. Lee, J. Kim, S. Kim, K. Kim, K. Yoo-Sung, and J. Park. Adaptation of a neighbor selection markov chain for prefetching tiled web gis data. In *Conference on Advances in Information Systems '02*.

[9] S. T. Leutenegger, M. A. Lopez, and J. Edgington. STR: a Simple and Efficient Algorithm for R-tree Packing. *ICDE '97*.

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

[11] P. Mishra and M. H. Eich. Join Processing in Relational Databases. *ACM Computing Surveys*, 24(1):63–113, 1992.

[12] S. Nobari, F. Tauheed, T. Heinis, P. Karras, S. Bressan, and A. Ailamaki. TOUCH: In-Memory Spatial Join by Hierarchical Data-Oriented Partitioning. In *SIGMOD '13*.

[13] D.-J. Park and H.-J. Kim. Prefetch Policies for Large Objects in a Web-enabled GIS Application. *Transactions on Data and Knowledge Engineering*, 37(1):65–84, 2001.

[14] T. K. Sellis, N. Roussopoulos, and C. Faloutsos. The R+-Tree: A Dynamic Index for Multi-Dimensional Objects. *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*.

[16] F. Tauheed, T. Heinis, F. Schürmann, H. Markram, and A. Ailamaki. SCOUT: Prefetching for Latent Structure Following Queries. In *VLDB '12*.