International Workshop on Visual Analytics (2012)
K. Matkovic and G. Santucci (Editors)

A System for Query Based Analysis and Visualization

Allen R. Sanderson[1], Brad Whitlock[2], Oliver Rübel[3], Hank Childs[3], Gunther Weber[3], Prabhat[3], and Kenseng Wu[3].

[1]Scientific Computing and Imaging Institute, University of Utah, Salt Lake City, USA
[2]Lawrence Livermore National Laboratory, Livermore, CA, USA
[3]Lawrence Berkeley National Laboratory, Berkeley, CA, USA

Abstract

Today scientists are producing large volumes of data that they wish to explore and visualize. In this paper we describe a system that combines range-based queries with fast lookup to allow a scientist to quickly and efficiently ask "what if?" questions. Unique to our system is the ability to perform "cumulative queries" that work on both an intra- and inter-time step basis. The results of such queries are visualized as frequency histograms and are the input for secondary queries, the results of which are then visualized.

Categories and Subject Descriptors (according to ACM CCS): I.6.6 [Simulation Output Analysis]: Visual analytics—Query based analysis

1. Introduction

In this paper we describe a system for exploration and visualization of large scale scientific data using query-based techniques that incorporate accelerated bitmap indexing. Queries on multidimensional data are a staple within the database community and are gaining prominence within the scientific community [GHA*08, GABJ08, SBC*06, RPW*08]. While previous work has typically focused on efficient queries for a single time step and utilizing those results at other time steps for data exploration, we focus on queries that require multiple time steps, the results of which are cumulative and can become the basis for additional queries. An example would be a change of state from one time step to another, i.e., an energetic particle that is either "passing" or "trapped."

Underlying our technique is the usage of bitmap indexing [WOS02] to accelerate data queries, which is important as the results can be returned on the order of seconds rather than minutes or hours with traditional techniques. We briefly describe this acceleration, followed by a discussion of the single time step queries which we build upon for our cumulative queries. We present an example from nuclear fusion, followed by a discussion, and concluding remarks.

2. Indexing Techniques

A common strategy to improve the accesses to large data sets is to develop auxiliary data structures known as indexes. A variety of indexes are available in database systems [OO00], many of which are variations of the B-Tree data structure which is designed to update quickly when the underlying data records are modified [Com79]. This feature is unnecessary for indexing scientific data sets unless the data is modified. For scientific data sets the bitmap index is a more appropriate indexing structure [O'N87, OW10].

2.1. Bitmap Indexing Technology

A bitmap index consists of a set of pairs of key value and a bitmap. The basic bitmap index uses one bitmap for each distinct key value with an entry of "1" if the key matches otherwise "0". With scientific data the number of distinct values can be as large as the number of entries (i.e., every value is distinct). In this case, the number of bits required to represent an index may scale quadratically with the number of entries, i.e., 10^9 entries may require 10^{18} bits and is much larger than the raw data size and is not acceptable.

A number of different strategies have been proposed to reduce the sizes of bitmap indexes and improve their overall effectiveness. We have chosen FastBit, which implements a number of techniques to improve the basic bitmap index including a compute-efficient compression [WOS02], multilevel encoding [WSS10], and clustering to improve binned bitmap indexes [WSS08]. FastBit has been shown to perform well in a number of scientific applications [WAB*09].

DOI: 10.2312/PE/EuroVAST/EuroVA12/025-029

2.2. Constructing the indexes

For the user to utilize FastBit we have created a lightweight library that produces the Fastbit bitmaps that are stored alongside the original data. When feasible, for point cloud data (i.e. particle data), such as high energy physics data, we take the extra step of sorting the entries based on their unique identifiers. This sorting further optimizes identifier-based queries which are used to reconstruct particle paths.

3. Single Time Step Queries

Single time step queries are a versatile tool for identification and extraction of temporally persistent data features. Temporal tracking and refinement of selections based on information from multiple time steps then supports detailed analysis of the temporal evolution of data features. Range-based queries form the foundation for defining data constraints to select data features of interest while identifier-based queries support persistent selection and tracking of features.

3.1. Identifier-based Queries

Identifier-based queries select data subsets based on their unique identifier (*id*) given by the local or global index of nodes or zones of a mesh or an explicit *id* variable. Explicit *id* variables are often used for data with highly dynamic topologies, such as particle data. Using *id*'s allows for explicit and consistent selection of data subsets via queries of the form $id \in \{10, 231, 501....\}$. In the context of the visualization system VisIt [CBB*05], *id*-based selections, also called Named Selections, are stored and managed by a central selection manager which allows us to link multiple visualizations and track selected data subsets over time. VisIt uses FastBit to efficiently evaluate the *id*-based queries to restrict a dataset to a given set of *id*'s using Named Selections.

3.2. Range-based Queries

Range-based queries are expressed in the context of scientific data as threshold ranges, i.e. $101 \leq x \leq 205$. In the context of multi-dimensional data, logical combinations of multiple range queries based on intersection (AND), union (OR) and negation (NOT) are often used. In practice, multi-dimensional queries of the form $(x < 100)$ *AND* $(y > 201)$, using only AND operations, are most commonly used because of their simplicity and intuitive use for selecting data subsets based on multiple constraints. In VisIt we can define *AND*-based multi-dimensional range queries explicitly as a series of data ranges or interactively using histogram-based parallel coordinates [RPW*08].

The range-based queries are processed using FastBit and can be directly translated to equivalent *id*-based queries managed by VisIt's central selection manager. Using this approach, we can track selections over time. By combining *id*-based and range-based queries, we can refine selections using range-criteria defined at different discrete points in time.

3.3. Accelerating Single Time Step Queries

Using bitmap indexing allows us to efficiently i) evaluate multi-dimensional range-based queries, ii) evaluate *id*-based queries and iii) compute conditional histograms. We integrate FastBit's indexing methods at the file-reader stage. VisIt's contract-based visualization pipeline permits different pipeline components, e.g., operator and plot filters, to include information about the data subsets, specified via *id*-based or range-based queries or conditional histograms.

Instead of reading the complete data set and performing the query, we use its index information and FastBit to perform the query. This allows us restrict the downstream processing to the data subsets, reducing the I/O footprint and computational complexity of the analysis. For example, VisIt's histogram-based parallel coordinates plot augments the contract so only the much smaller histograms required for rendering are communicated by the file reader instead of large amounts of raw data [RPW*08].

4. Cumulative Queries

Cumulative queries extend the query-driven analysis to incorporate information from the complete time series and enable identification of temporal features that cannot be seen within single discrete time steps. The temporal features may be identified via intra- or inter-time step queries. For instance, how frequently does a particle's weight exceed a value at all time steps (intra-time) versus how frequently does a particle change state from "passing" to "trapped" going from one time step to the next (inter-time), Equation 1.

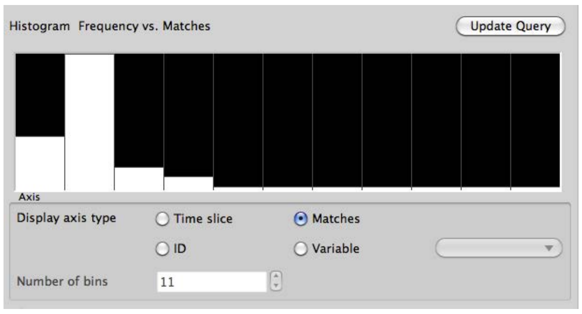
$$(wt_t > 0)\ AND\ (trapped_t\ NE\ trapped_{t+1}) \qquad (1)$$

Similar to single time step query results, cumulative query results can be used directly for data selection simply by applying a boolean operator to convert the frequency counts to true or false. This step effectively turns the result into the following; return all entries that meet the query at least once during the simulation. While such queries are of interest, there is a significant amount of information that can be further mined by refining the cumulative query results via secondary queries.

4.1. Secondary Queries

Secondary queries can be based on multiple criteria. Here we highlight three such criteria using histograms which allows the user to sub-select entries based on their associated frequency. For instance, after performing the initial cumulative query over 50 time steps (Equation 1) the user is presented a histogram based on the distribution of query matches and their frequency (Figure 1), thus allowing the user to choose the entries based on the frequency of matches.

Another histogram that can be presented to the user is based on the frequency of matches on a per time step basis, highlighting trends over time or instantaneous spikes at particular time steps, Figure 2. The last histogram presented uses data values that may or may not have been used in the query but who's entries matched the query. Again, this allows the user to look for correlations and trends in the data from those entries that matched the query, Figure 3. All of the histogram choices are analytical tools that allow visual exploration as part of the cumulative query process.

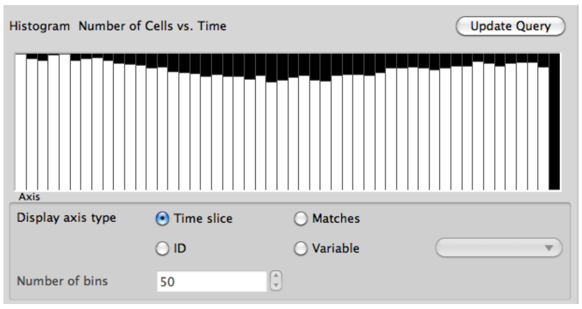

Histogram Frequency vs. Matches

Update Query

Display axis type: ( ) Time slice, [✓ Matches], ( ) ID, ( ) Variable

Number of bins: 11

Figure 1: *A histogram showing the distribution and frequency of query matches. Each bin represents the number of matches to the query with the left most bin representing those particles matching the query 1 out of 50 time steps and the right most bin representing those particles matching the query 11 out of 50 time steps. The majority of particles entries matched 2 time steps while no particles matched 12 or more time steps.*


Histogram Number of Cells vs. Time

Update Query

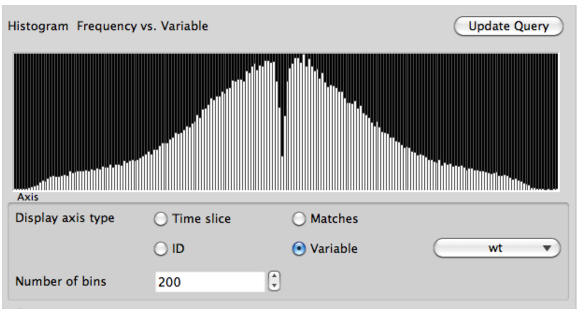
Display axis type: [✓ Time slice], ( ) Matches, ( ) ID, ( ) Variable

Number of bins: 50

Figure 2: *A histogram showing the frequency of matches for all 50 time steps. Each bin represents a time step and the number of particles that matched the query. Fewer particles matched the query at time steps in the middle of the simulation than at the beginning and end indicating that these time steps may be of significance to the application scientist.*

4.2. Cumulative query process

Cumulative queries are implemented in VisIt and extend its concept of the Named Selection, which restricts datasets to a given set of *id*'s. Cumulative query selections are realized by adding three filters to the visualization pipeline within


Histogram Frequency vs. Variable

Update Query

Display axis type: ( ) Time slice, ( ) Matches, ( ) ID, [✓ Variable] wt

Number of bins: 200

Figure 3: *A histogram showing the distribution of particle weights from entries that match the initial query. The frequency reflects a Gaussian distribution except near the middle where the weight is near zero. Those particles with high weights would be of interest to the application scientist.*

VisIt's compute engine, Figure 4. These filters accomplish the cumulative query's functions of range queries over time, followed by secondary queries that let the user further subselect the set of *id*'s that become part of the final selection. First, a filter is added to obtain histogram information from FastBit for the selection summary shown in the VisIt GUI. The output of the histogram filter is sent into a threshold filter that issues range queries to FastBit, resulting in a reduced dataset. The output of the threshold filter is passed into the cumulative query filter. This filter executes the pipeline for a range of time steps, creating datasets for which the range queries hold true over all time. Once all time steps have produced an output, the filter iterates over them and obtains the frequency each *id* is present. The frequencies are cached so full re-executions of the filter are not required when the user changes query parameters that pertain to secondary queries.

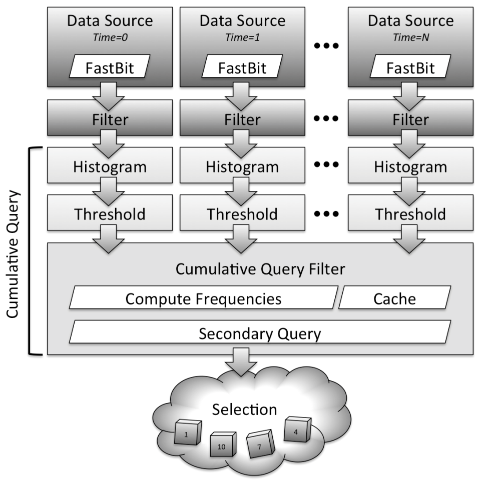
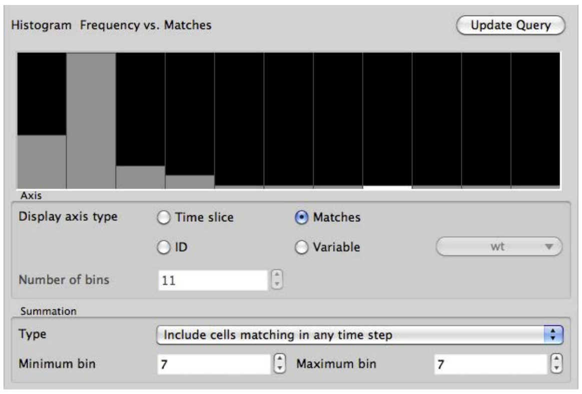


Figure 4: *A schematic of the cumulative query pipeline in VisIt. Each column represents the query for one time step.*

The secondary query begins by applying a summation rule to the calculated frequencies. The summation rule selects *id*'s that are present in any time step or *id*'s that are present in all time steps to produce another set of *id*'s that will be processed further. Those *id*'s are sorted on the basis of frequency, matches, *id*, or associated variable value. The sorted *id*'s are then grouped into a user-selected number of bins and a range of bins is selected by the user. The *id*'s contained in the selected bins are retrived and summary information describing the query is created for the user so they may further refine the cumulative query parameters. At this point, VisIt can apply the selection to any of its plots in order to visualize and analyze only the selected data subsets.

5. Results and Discussion

We now apply the cumulative query tool to look at particle-in-cell data from a gyrokinetic simulation of the turbulence in a magnetically confined fusion experiment [ZLW00]. For this query we are interested in particles that changed from a "passing" to a "trapped" state during the simulation. This initial query resulted in over 120K out of 500K particles having at least one state change over the course of 50 time steps. From Figure 1 we find that one particle changed state eleven times. For our example, we perform a secondary query, choosing only the seventh bin which contains only those particles that changed state seven times, Figure 5. This secondary query results in eight particles being selected.



Display axis type: Time slice [✓ Matches] ID Variable wt

Number of bins: 11

Type: Include cells matching in any time step

Minimum bin: 7 Maximum bin: 7

Figure 5: *The histogram from Figure 1 showing the distribution of matches along with the summation at the bottom which performs our secondary query. Here bin 7 was subselected yielding eight particles that matched the query at 7 out of 50 time steps. Note that in contrast to Figure 1 all of the bins but bin 7 are grayed out.*

From the selection, an *id*-based query is performed to extract the eight particles' temporally persistent positional data and instantaneous trapped state. The positional data is used to construct the particles' paths while the trapped state is used for color. Using FastBit, finding the data for these eight out of 500K particles over 50 times steps takes tens of seconds, whereas without FastBit, the same operations takes several minutes. While these are just preliminary findings it does demonstrate the acceleration provided by the bitmap indexing. Additional, more formal measurements for single time operations can be found in [RPW*08]. In Figure 6 we show the resulting paths of the eight particles where it is possible to discern a burst-like occurrence in the state changes. Such burst-like state changes cannot be detected via the query process and shows the utility of combining the query process, with FastBit indexing, and visual exploration.

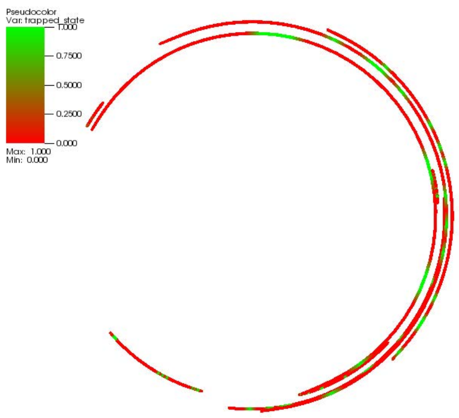


Figure 6: *Particle paths from the secondary query showing the trapped (red) and passing (green) states. Though each particle changed state seven times, occuring in bursts of two or three followed by periods of stability, the changes appear to be independent of position, velocity, or time step.*

6. Conclusions

In this paper we have extended our query-based analysis to include cumulative queries that operate on an intra- and inter-time step basis. Such queries capture temporal features in the data that can not be extracted via single time step queries. The queries have been deployed as part of the VisIt visualization and analysis system.

Our future work is to expand the analytical queries to include geometric queries on the particle paths. These queries would require an on-the-fly moving window where the last time step is dropped while adding the next to perform the geometric query. Further, as data continues to grow it will be necessary to parallelize singe time step and cumulative queries in order to maintain reasonable response times.

7. Acknowledgments

This research was funded in part by the United States Department of Energy's Visualization and Analytics Center for Enabling Technology. The authors thank Stephane Ethier at PPPL for the gyrokentic data.

References

[CBB*05] CHILDS H., BRUGGER E. S., BONNELL K. S., MEREDITH J. S., MILLER M., WHITLOCK B. J., MAX N.: A contract-based system for large data visualization. In *Proceedings of IEEE Visualization 2005* (2005), pp. 190–198. 2

[Com79] COMER D.: The ubiquitous B-tree. *Computing Surveys 11*, 2 (1979), 121–137. 1

[GABJ08] GOSINK L. J., ANDERSON J. C., BETHEL E. W., JOY K. I.: Query-driven visualization of time-varying adaptive mesh refinement data. *IEEE Transactions on Visualization and Computer Graphics 14*, 6 (Nov. 2008), 1715–1722. 1

[GHA*08] GLATTER M., HUANG J., AHERN S., DANIEL J., LU A.: Visualizing temporal patterns in large multivariate data using modified globbing. *IEEE Transactions on Visualization and Computer Graphics 14*, 6 (Nov. 2008), 1467–1474. 1

[O'N87] O'NEIL P.: Model 204 architecture and performance. In *2nd International Workshop in High Performance Transaction Systems, Asilomar, CA* (Sept. 1987), vol. 359 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 40–59. 1

[OO00] O'NEIL P., O'NEIL E.: *Database: principles, programming, and performance*, 2nd ed. Morgan Kaugmann, 2000. 1

[OW10] OTOO E., WU K.: *Accelerating Queries on Very Large Datasets*. Chapman & Hall/CRC Computational Science, 2010, ch. 6, pp. 183–234. 1

[RPW*08] RÜBEL O., PRABHAT, WU K., CHILDS H., MEREDITH J., GEDDES C. G. R., CORMIER-MICHEL E., AHERN S., WEBER G. H., MESSMER P., HAGEN H., HAMANN B., BETHEL E. W.: High performance multivariate visual data exploration for extremely large data. In *Proceedings of the 2008 ACM/IEEE conference on Supercomputing* (Piscataway, NJ, USA, 2008), SC '08, IEEE Press, pp. 51:1–51:12. 1, 2, 4

[SBC*06] STOCKINGER K., BETHEL E. W., CAMPBELL S., DART E., WU K.: Detecting distributed scans using high-performance query-driven visualization. In *Proceedings of the 2006 ACM/IEEE conference on Supercomputing* (New York, NY, USA, 2006), SC '06, ACM. 1

[WAB*09] WU K., AHERN S., BETHEL E. W., CHEN J., CHILDS H., CORMIER-MICHEL E., GEDDES C., GU J., HAGEN H., HAMANN B., KOEGLER W., LAURET J., MEREDITH J., MESSMER P., OTOO E., PEREVOZTCHIKOV V., POSKANZER A., PRABHAT, RUBEL O., SHOSHANI A., SIM A., STOCKINGER K., WEBER G., ZHANG W.-M.: FastBit: Interactively searching massive data. In *SciDAC* (2009). 1

[WOS02] WU K., OTOO E., SHOSHANI A.: Compressing bitmap indexes for faster search operations. In *Proceedings of SSDBM'02* (Edinburgh, Scotland, 2002), pp. 99–108. LBNL-49627. 1

[WSS08] WU K., STOCKINGER K., SHOSANI A.: Breaking the curse of cardinality on bitmap indexes. In *SSDBM'08* (2008), Springer, pp. 348–365. Preprint appeared as LBNL Tech Report LBNL-173E. 1

[WSS10] WU K., SHOSHANI A., STOCKINGER K.: Analyses of multi-level and multi-component compressed bitmap indexes. *ACM Transactions on Database Systems* (2010), 1–52. 1

[ZLW00] Z. LIN T. S. HAHM W. W. L. W. M. T., WHITE R. B.: Gyrokinetic simulations in general geometry and applications to collisional damping of zonal flows. *Phyics of Plasmas 7*, 5 (2000), 1857–1862. 4