Lawrence Berkeley National Laboratory
Lawrence Berkeley National Laboratory

Title
Query-Driven Visualization and Analysis

Permalink
https://escholarship.org/uc/item/2xq2d9d5

Author
Ruebel, Oliver

Publication Date
2012-11-01

eScholarship.org Powered by the California Digital Library
University of California

Query-Driven Visualization and Analysis

Oliver Rübel

Computational Research Division
Lawrence Berkeley National Laboratory,
Berkeley, California, USA, 94720.

E. Wes Bethel

Computational Research Division
Lawrence Berkeley National Laboratory,
Berkeley, California, USA, 94720.

Prabhat

Computational Research Division
Lawrence Berkeley National Laboratory,
Berkeley, California, USA, 94720.

Kesheng Wu

Computational Research Division
Lawrence Berkeley National Laboratory,
Berkeley, California, USA, 94720.

October 2012

Acknowledgment

This work was supported by the Director, Office of Science, Office and Advanced Scientific Computing Research, of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231. Some of the research in this work used resources of the National Energy Research Scientific Computing Center, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.

Legal Disclaimer

This document was prepared as an account of work sponsored by the United States Government. While this document is believed to contain correct information, neither the United States Government nor any agency thereof, nor The Regents of the University of California, nor any of their employees, makes any warranty, express or implied, or assumes any legal responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by its trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government or any agency thereof, or The Regents of the University of California. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof or The Regents of the University of California.

Abstract

This report focuses on an approach to high performance visualization and analysis, termed *query-driven visualization and analysis* (QDV). QDV aims to reduce the amount of data that needs to be processed by the visualization, analysis, and rendering pipelines. The goal of the data reduction process is to separate out data that is "scientifically interesting" and to focus visualization, analysis, and rendering on that interesting subset. The premise is that for any given visualization or analysis task, the data subset of interest is much smaller than the larger, complete data set. This strategy—extracting smaller data subsets of interest and focusing of the visualization processing on these subsets—is complementary to the approach of increasing the capacity of the visualization, analysis, and rendering pipelines through parallelism. This report discusses the fundamental concepts in QDV, their relationship to different stages in the visualization and analysis pipelines, and presents QDV's application to problems in diverse areas, ranging from forensic cybersecurity to high energy physics.

Preface

The material in this technical report is a chapter from the book entitled *High Performance Visualization—Enabling Extreme Scale Scientific Insight* [4], published by Taylor & Francis, and part of the CRC Computational Science series.

Contents

1 Introduction

Query-driven visualization and analysis (QDV) addresses the big-data analysis challenge by focusing on data that is "scientifically interesting" and, hence, reducing the amount of data that needs to be processed by the visualization, analysis, and rendering pipelines. The data reduction is typically based on range queries, which are used to constrain the data variables of interest. The term QDV was coined by Stockinger et al. [31, 32] to describe the combination of high performance index/query methods with visual data exploration methods. Using this approach, the complexity of the visualization, analysis and rendering is reduced to $O(k)$, with k being the number of objects retrieved by the query. The premise of QDV, then, is that for any given visualization or analysis task, the data subset of interest, k, is much smaller than the larger, complete data set. QDV methods are among a small subset of techniques that are able to address both large *and* highly complex data.

In a typical QDV task, the user begins the analysis by forming definitions for data subsets of interest. This characterization is done through the concept of range queries, that is, the user defines a set of constraints for variables of interest. For example, in the context of a combustion simulation, a scientist may only be interested in regions of a particular temperature t and pressure p, such that ($1000\text{F} < t < 1500\text{F}$) AND ($p < 800\text{Pa}$). QDV uses such range queries to filter data before it is passed to the subsequent data processing pipelines, thereby, focusing the visualization and analysis exclusively on the data that is meaningful to the user. In practice, a user may not have a precise notion of which parts of the data are of the most interest. Rather than specifying the exact data subset(s) of interest, a user may begin the analysis by excluding large data portions that are known to be of no interest and/or specifying regions that are known to contain the data of interest. Through a process of iterative refinement of queries and analysis of query results, the user is able to gain further insight into the data and identify and specify more precisely the data subset(s) of interest.

QDV inherently relies on two main technologies: fast methods for evaluating data queries; and efficient methods for specification, validation, visualization, and analysis of query results. In the context of QDV of scientific data, approaches based on bitmap indexing have shown to be very effective for accelerating multivariate data queries, which will be the focus of Section 2. Then, Section 3 discusses different methods used for specification, validation, and visualization of data queries, with a particular focus on parallel coordinates as an effective interface for formulating multivariate data queries (see Section 3.1). To facilitate the query-based data analysis process, the QDV approach has also been combined with automated analysis methods to suggest further query refinements, as well as to automate the definition of queries and extraction of features of interest. As an example, Section 3.2 discusses various methods for the segmentation and labeling of query components. To illustrate the effectiveness and wide applicability of QDV methods, two case studies are investigated: the applications of QDV to network traffic data in Section 4.1, and particle accelerator simulation data in Section 4.2. This chapter concludes with an evaluation of the QDV approach and a discussion of future challenges in Section 5.

2 Data Subsetting and Performance

Efficient methods for data subsetting are fundamental to the concept of QDV. The database community has developed a variety of indexing methods to accelerate data searches. Many popular database systems use variations of the B-tree [2]. The B-tree was designed for transaction-type applications, exemplified by interactions between a bank and its customer. Interactions with transactional data are characterized by the frequent modification of data records, one record at a time, and retrieval of a relatively small number of data records, such as the look-up of a customer's banking information. In contrast, scientific data is typically not modified after creation, but is processed in a read-only fashion. Furthermore, scientific search operations typically retrieve many more data records than typical transaction-type queries, and the number of records may also vary considerably more. For example, when studying the ignition process in a combustion data set, a query that isolates regions of high temperature may initially only retrieve very few data records; however, as the combustion process progresses from an initial spark into a large flame, the same query may result in the retrieval of a sizable fraction of the data. Scientific search operations also often involve a large number of data dimensions. For example, when analyzing the combustion process, a scientist may be interested in regions that exhibit high temperature, high pressure, and contain a high/low concentration of different chemical species. For these types of versatile, high-dimensional, read-only query operations, the bitmap index is a more appropriate indexing structure [29], which is the subject of Section 2.1.

There is a pressing need for efficient data interfaces that make indexing technology accessible to the scientific data processing pipelines. Across scientific applications, array-based data models are most commonly used for data storage. However, common array-based scientific data formats, such as HDF5 [33] and NetCDF [34], do not support semantic indexing methods. A number of array-based research databases, such as SciDB [7] or MonetDB [5], provide advanced index/query technology, are optimized for scientific data. However, scientific data is commonly stored outside of a database system. In practice, it is often too expensive to make copies of extremely large scientific data sets for the purpose of indexing, and the conversion of data to different file formats may break downstream data processing pipelines. Query-based analyses, hence, require efficient data interfaces that make semantic indexing methods accessible within state-of-the-art scientific data formats (see Section 2.2).

The first section, Section 2.1, discusses bitmap indexing in detail. Then, in Section 2.2, FastQuery [10] is introduced as an example index and query system for integrating bitmap indexing technology with state-of-the-art scientific data formats, including extensions of bitmap indexing to the processing of queries on large distributed-memory platforms.

2.1 Bitmap Indexing

A bitmap index stores the bulk of its information in a series of bit sequences known as bitmaps. Figure 1 shows a small example with one variable named **I**, with a value range of 0, 1, 2, or 3. For each of the four possible values, a separate bitmap is created to record which rows, or records, contain the corresponding value. For example, the value 0 only appears in row 1. The bitmap representing the value 0, hence, contains a 1, as the first bit, while the remaining bits are 0. This is the most basic form of a bitmap index [23].

A scientific data set often involves many variables, and a query may involve an arbitrary combination of the variables. When the B-Tree index [11] or a multidimensional index [14] is used, the most efficient way to answer a query is to create an index with the variables involved in the query. This approach requires a different index for each combination of variables. Because the number of possible combinations is large, a large amount of disk space is needed to store the different indices. These combinations are necessary because the results from these tree-based indices cannot be easily combined to produce the final answer. In contrast, when using a bitmap index to answer a query, a bitmap is produced to represent the answer. A multivariate query can be answered by resolving the condition on each variable separately, using the bitmap index for that variable, and then combining all the intermediate answers with bitwise logical operations to produce the final answer. In this case, the total size of all bitmap indices grows linearly with the number of variables in the data. Typically, the bitwise logical operations are well-supported by computer hardware. Therefore, the bitmap indices can answer queries efficiently.

The variable **I** shown in Figure 1 has only four possible values. In general, for a variable with C distinct values, the basic bitmap index needs C bitmaps of N bits each, where N is the number of rows (or records) in the data set [9]. If C is the cardinality of the variable, then in a worst-case scenario, $C = N$, and the corresponding bitmap index has N^2 bits. Even for a moderate size data set, N^2 bits can easily fill up all disks of a large computer system. Ideally, the index size should be proportional to N, instead of N^2. The techniques for controlling the bitmap index sizes roughly fall in three categories: compression, encoding, and binning.

Compression: In principle, any lossless compression method can be used to compress bitmap indices. In practice, the most efficient bitmap compression methods are based on run-length

		Bitmap Index			
RID	**I**	=0	=1	=2	=3
1	0	1	0	0	0
2	1	0	1	0	0
3	2	0	0	1	0
4	3	0	0	0	1
5	3	0	0	0	1
6	3	0	0	0	1
7	3	0	0	0	1
8	1	0	1	0	0
		b_1	b_2	b_3	b_4

Figure 1: A sample bitmap index where RID is the record ID and **I** is the integer attribute with values in the range of 0 to 3.

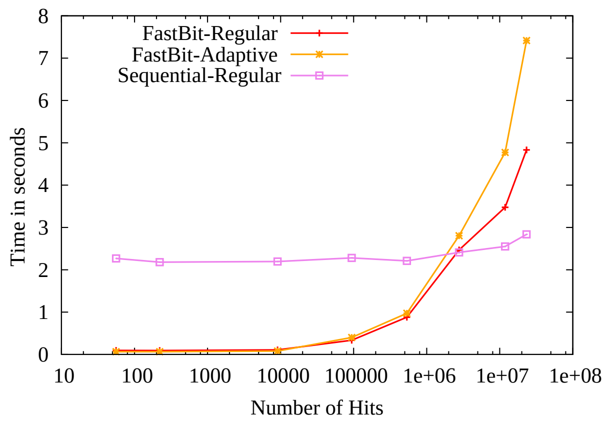
encoding, as they support bitwise logical operations on the compressed data without decompressing the bitmaps [1, 38]. Working directly with the compressed bitmaps reduces the time needed for reading the bitmaps from disk to memory, as well as the amount of time to perform the necessary computations. Furthermore, using Word-Aligned Hybrid (WAH) compression, it was shown that the time needed to evaluate a query is, in the worst case, a linear function of the number of records h that satisfy the query [39]. Therefore, the WAH compressed bitmap index has an optimal computational complexity of $O(h)$. In the context of QDV, this means that the computational complexity of the analysis no longer depends on the total of number of data records, N, but only on the number of records, h, that define the feature(s) of interest defined by the query.

Encoding: The basic bitmap encoding used in Figure 1 is also called *equality-encoding*, as it shows the best performance for *equality queries*, such as temperature, $t = 100$F. Other well-known bitmap encoding methods include the *range encoding* [8] and the *interval encoding* [9]. They are better suited for other query types, such as (35.8Pa $< p <$ 56.7Pa). There are also a number of different ways of composing more complex bitmap indices by using combinations of the equality, range, and interval encoding [41].

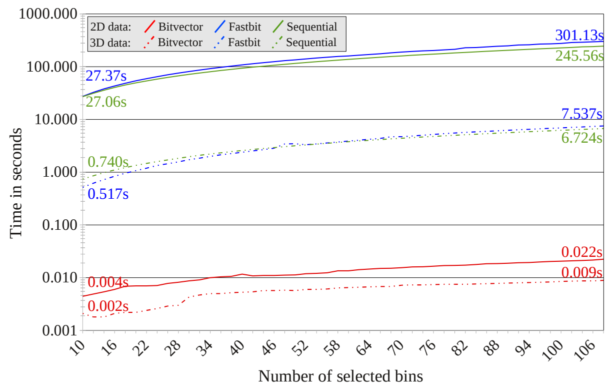
Binning: The basic bitmap index works well for low-cardinality attributes, such as "gender," "types of cars sold per month," or "airplane models produced by Airbus and Boeing." However, for scientific data, the variables often have a large value-range, while different values are hardly ever repeated. These types of scientific data are referred to as high-cardinality data. A bitmap index for a high-cardinality variable typically requires many bitmaps. One can reduce the number of bitmaps needed by binning the data. However, the corresponding index would not be able to answer some queries accurately. The cost of resolving the query accurately could be quite high [26]. Additional data structures might be needed to answer the queries with predictable performance [43].

The bitmaps inside the bitmap indices also offer a way to count the number of records satisfying certain conditions quickly. This feature can be used to quickly compute *conditional histograms*. For example, in addition to directly counting from the bitmaps, it is also possible to count the rows in each histogram bucket by reading the relevant raw data, or use a combination of bitmaps and the raw data. Stockinger et al. [30] described a set of algorithms for efficient parallel computation of conditional histograms. The ability to quickly compute conditional histograms, in turn, accelerates many histogram-based visualization methods, such as histogram-based parallel coordinates described later in Section 3.1. Figure 2a illustrates the serial performance for computing conditional histograms, using bitmap indexing by way of the FastBit [37] software.

Histogram-based analysis methods—such as density-based segmentation and feature detec-



(a) Conditional histogram performance.



(b) Bin-query performance.

Figure 2: On the left, (a) shows timings for serial computation of regularly and adaptively binned 2D histograms on a 3D ($\approx 90 \times 10^6$ particles) particle data set using bitmap indexing and a baseline sequential scan method. Image source: Rübel et al. 2008 [28]. Timings for serial evaluation of 3D bin-queries are shown in (b), using a 2D ($\approx 2.4 \times 10^6$ particles) and 3D ($\approx 90 \times 10^6$ particles) particle data set. Bin queries are evaluated using: (1) per-bin bitvectors returned by FastBit, (2) FastBit queries, and (3) a baseline sequential scan method. Image source: Rübel et al., 2010 [27].

tion methods—require the ability to evaluate bin-queries efficiently [27]. A *bin-query* extracts the data associated with a set of histogram bins and is comprised of a series of queries, for example, in the 3D case, of the form $[(x_i \leq x < x_{i+1}) AND (y_i \leq y < y_{i+1}) AND (z_i \leq z < z_{i+1})]$, where i indicates the index of a selected bin and x, y, and z refer to the dimensions of the histogram. Instead of evaluating complex bin-queries explicitly, one can use bitmaps—one for each nonzero histogram bin—to efficiently store the inverse mapping from histogram-bins to the data. Figure 2b illustrates the performance advantage of this approach for evaluating bin-queries. In practice, the overhead for computing the per bin bitmaps depends on the number of nonzero bins, but is, in general, moderate.

A number of the above described methods have been implemented in an open-source software called FastBit [37]. In the context of QDV, FastBit is used to process range queries and equality queries, as well as to compute conditional histograms and bin-queries. FastBit has also been integrated with the parallel visualization system VisIt, making FastBit-based QDV capabilities available to the user community.

2.2 Data Interfaces

In order to make effective use of semantic indexing methods to accelerate data subselection, advanced data interfaces are needed that make index and query methods accessible within state-of-the-art scientific data formats. Such interfaces should ideally have the following characteristics. First, they enable access to a large range of scientific data formats. Second, they avoid costly data copy and file conversion operations for indexing purposes. Third, they scale to massive data sets. Fourth, they are capable of performing indexing and query operations on large, distributed, multicore platforms. For example, Gosink et al. [17] described HDF5-FastQuery, a library that integrates serial index/query operations using FastBit with HDF5.

Recently, Chou et al. introduced FastQuery [10], which integrates parallel-capable index/query operations using FastBit—including bitmap index computation, storage, and data subset selection—with a variety of array-based data formats. FastQuery provides a simple array-based I/O interface to the data. Access to the data is then performed via data format specific readers that implement the FastQuery I/O interface. The system is, in this way, agnostic to the underlying data format and can be easily extended to support new data formats. FastQuery also uses FastBit for index/query operations. To enable parallel index/query operations, FastQuery uses the concept of subarrays. Similar to Fortran and other programming languages, subarrays are specified using the general form of *lower : upper : stride*. Usage of subarrays provides added flexibility in the data analysis, but more importantly—since subarrays are, by definition, smaller than the complete data set—index/query times can be greatly reduced compared to approaches that are constrained to processing the entire data set. This subarray feature, furthermore, enables FastQuery to divide the data into chunks during index creation, chunks that can be processed in parallel in a distributed-memory environment. Evaluation of data queries is then parallelized in a similar fashion. In addition to subarrays, FastQuery also supports parallelism across files and data variables. Chou et al. demonstrated good scalability of both indexing and query evaluation, to several thousands of cores. For details, see the work by Chou et al. [10].

3 Formulating Multivariate Queries

Semantic indexing provides the user with the ability to quickly locate data subsets of interest. In order to make effective use of this ability, efficient interfaces and visualization methods are needed that allow the user to quickly identify data portions of interest, specify multivariate data queries to extract the relevant data, and validate query results. As mentioned earlier, a QDV-based analysis is typically performed in a process of iterative refinement of queries and analysis of query results. To effectively support such an iterative workflow, the query interface and visualization should provide the user with feedback on possible strategies to refine and improve the query specification, and be efficient to provide the user with fast, in-time feedback about query-results, in particular, within the context of large data.

Scientific visualization is very effective for the analysis of physical phenomena and plays an important role in the context of QDV for the validation of query results. Highlighting query results in scientific visualizations provides an effective means for the analysis of spatial structures and distributions of the selected data portions. However, scientific visualization methods are limited with respect to the visualization of high-dimensional variable space in that only a limited number of data dimensions can be visualized at once. Scientific visualizations, hence, play only a limited role as interfaces for formulating multivariate queries.

On the other hand, information visualization methods—such as scatter-plot matrix and parallel coordinate plots—are very effective for the visualization and exploration of high-dimensional variable spaces and the analysis of relationships between different data dimensions. In the context of QDV, information visualizations, therefore, play a key role as interfaces for the specification of complex, multidimensional queries and the validation of query results.

Both scientific and information visualization play an important role in QDV. In the context of QDV, multiple scientific and information visualization views—each highlighting different aspects of the data—are, therefore, often linked to highlight the same data subsets (queries) in a well-defined manner to facilitate effective coordination between the views. In literature, this design pattern is often referred to as brushing and linking [35]. Using multiple views allows the user to analyze different data aspects without being overwhelmed by the high dimensionality of the data.

To ease validation and refinement of data queries, automated analysis methods may be used for the post-processing of query results to, for example, segment and label the distinct spatial components of a query. Information derived through the post-processing of query results provides important means to enhance the visualization, to help suggest further query refinements, and to automate the definition of queries to extract features of interest.

The next sections describe the use of parallel coordinates as an effective interface to formulate high-dimensional data queries (see Section 3.1). Afterwards, the post-processing of query results are discussed, enhancing the QDV-based analysis through the use of automated methods for the segmentation of query results and methods for investigation of the importance of variables to the query solution and their interactions (see Section 3.2).

3.1 Parallel Coordinates Multivariate Query Interface

Parallel coordinates are a common information visualization technique [19]. Each data variable is represented by a vertical axis in the plot (see Fig. 3). A parallel coordinates plot is constructed by drawing a polyline connecting the points where a data record's variable values intersect each axis.

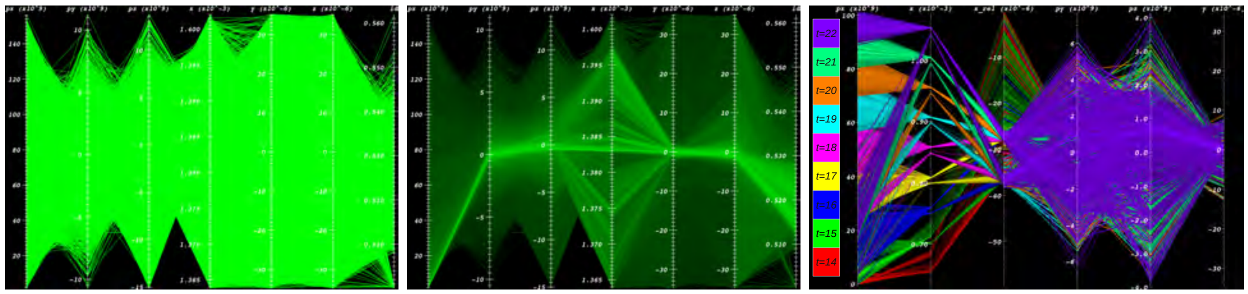
Parallel coordinates provide a very effective interface for defining multidimensional range queries. Using sliders attached to each parallel axis of the plot, a user defines range thresholds in each displayed data dimension. Selection is performed iteratively by defining and refining thresholds one axis at a time. By rendering the user-selected data subset (the focus view) in front of the parallel coordinates plot created from the entire data set—or a subset of the data defined using a previous query—(the context view), the user receives immediate feedback about general properties of the selection (see Fig. 7). Data outliers stand out visually as single or small groups of lines diverging from the main data trends. Data trends appear as dense groups of lines (or bright colored bins, in the case of histogram-based parallel coordinates). A comparison of the focus and context view helps to convey understanding about similarities and differences between the two views. Analysis of data queries defined in parallel coordinates, using additional linked visualizations—such as physical views of the data—then provides additional information about the structure of a query. These various forms of visual query feedback help to validate and refine query-based selections. Figure 7 exemplifies the use of parallel coordinates for the interactive query-driven analysis of laser plasma particle acceleration data discussed in more detail later in Section 4.2.2.

Parallel coordinates also support iterative, multiresolution exploration of data at multiple time-scales. For example, initially a user may view data on a weekly scale. After selecting a week(s) of interest, this initial selection may be used as context view. By scaling the parallel axis to show only the data ranges covered by the context selection—also called dimensional scaling—the selection can be viewed and refined in greater detail at daily resolution.

In practice, parallel coordinates have disadvantages when applied to very large data. In the traditional approach, the parallel coordinates plot is rendered by drawing a polyline for each data record. When there are relatively few data records, this approach is reasonable and produces legible results. But when applied to large data sets, the plot can be quickly cluttered and difficult to interpret, as is the case in Figure 3a. Worst of all, the computational and rendering complexity of parallel coordinates is proportional to the size of the data set. As data sizes grow, these problems quickly become intractable.

Histogram-based parallel coordinates [22, 28] are an efficient, density-based method for computing and rendering parallel coordinates plots. Rather than viewing the parallel coordinates plot as a collection of polylines (one polyline per data record), one can discretize the relationship of all data records between pairs of parallel axes using 2D histograms.

Based on the 2D histograms—one per neighboring axes pair—rendering proceeds by drawing



(a) (b) (c)

Figure 3: The left images, (a) and (b), show a comparison of two different parallel coordinate renderings of a particle data set consisting of 256,463 data records and 7 variables using: (a) traditional line-based parallel coordinates and (b) high-resolution, histogram-based parallel coordinates with 700 bins per data dimension. The histogram-based rendering reveals many more details when displaying large numbers of data records. Image (c) shows the temporal histogram-based parallel coordinates of two particle beams in a laser-plasma accelerator data set, at timesteps $t = [14, 22]$. Color is used to indicate the discrete timesteps. The two different beams can be readily identified in x (second axis). Differences in the acceleration can be clearly seen in the momentum in the x direction, px (first axis). Image source: Rübel et. al, 2008 [28].

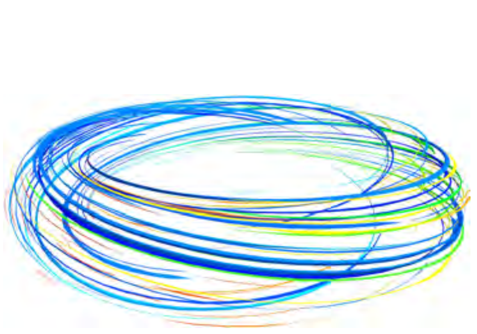
one quadrilateral per nonempty bin, where each quadrilateral connects two data ranges between the neighboring axes. Quadrilateral color and opacity is a function of histogram bin magnitude, so more densely populated regions are visually differentiated from regions with lower density. This approach has a significant advantage for large data applications: the rendering complexity no longer depends on the size of the original data, but only on the resolution of the underlying histograms. The histograms, for the context view, need to be computed only once. The calculation of the focus view histograms happens in response to user changes to query ranges and are accelerated by FastBit. Previous studies examine the scalability characteristics of those algorithms on large supercomputers when applied to very large scientific data sets [3, 28, 30].

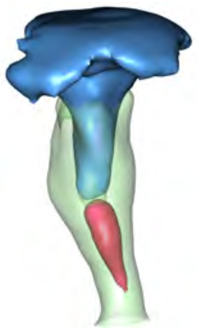
Figure 3b shows how more information in data is revealed through a combination of visualization and rendering principles and techniques. For example, color brightness or transparency can convey the number of records per bin, and rendering order takes into account region density so that important regions are not hidden by occlusion. Different colors could be assigned to different timesteps, producing a temporal parallel coordinates plot (see Fig. 3c). This approach is useful in helping to reveal multivariate trends and outliers across large, time-varying data sets.

3.2 Segmenting Query Results

A query describes a binary classification of the data, based on whether a record satisfies the query condition(s). However, typically, the feature(s) of interest to the end users are not individual data records, but regions of space defined by connected components of a query—such as ignition kernels or flame fronts. Besides physical components of a query, a feature of interest may also be defined by groups of records (clusters) in high-dimensional variable space. Combining QDV with methods for the segmentation and classification of query results can facilitate the analysis of large data sets by supporting the identification of subfeatures of a query, by suggesting strategies for the refinement of queries, or by automating the definition of complex queries for automatic feature detection.

A common approach for enhancing the QDV analysis process consists of identifying spatially connected components in the query results. This has been accomplished in the past using a technique known as connected component labeling [36, 40]. Information from connected component


(a)


(b)

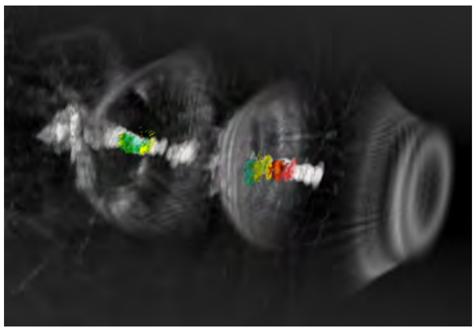

(c)

Figure 4: Applications of segmentation of query results. Image (a) displays the magnetic confinement fusion visualization showing regions of high magnetic potential colored by their connected component label. Image source: Wu et. al, 2011 [42]. Image (b) shows the query selecting the eye of a hurricane. Multivariate statistics-based segmentation reveals three distinct regions in which the query's joint distribution is dominated by the influence of pressure (blue), velocity (green), and temperature (red). Image source: Gosink et. al, 2011 [18]. Image (c) is the volume rendering of the plasma density (gray), illustrating the wake of the laser in a plasma-based accelerator. The data set contains $\approx 229 \times 10^6$ particles per timestep. The particles of the two main beams, automatically detected by the query-based analysis, are shown colored by their momentum in acceleration direction (px). Image source: Rübel et. el, 2009 [27].

labels provides the means to enhance the visualization—as shown in Figure 4b—and enables further quantitative analysis. For example, statistical analysis of the number and distributions in size or volume of physical components of query results can provide valuable information about the state and evolution of dynamic physical processes, such as a flame [6].

Connected component analysis for QDV has a wide range of applications. Stockinger et al. [32] applied this approach to combustion simulation data. In this context, a researcher might be interested in finding ignition kernels, or regions of extinction. On the other hand, when studying the stability of magnetic confinement for fusion, a researcher might be interested in regions with high electric potential because of their association with zonal flows, critical to the stability of the magnetic confinement—as shown in Figure 4a [42].

In practice, connected component analysis is mainly useful for data with known topology, in particular, data defined on regular meshes. For scattered data—such as particle data—where no connectivity is given, density-based clustering approaches may provide a useful alternative to group selected particles based on their spatial distribution. One of the main limitations of a connected component analysis, in the context of QDV, is that the labeling of components by itself does not yield any direct feedback about possible strategies for the refinement of data queries.

To address this problem, Gosink et al. [18] proposed the use of multivariate statistics to support the exploration of the solution space of data queries. To analyze the structure of a query result, they used kernel density estimation to compute the joint and univariate probability distributions for the multivariate solution space of a query. Visual exploration of the joint density function—for example, using isosurfaces in physical space—helps users with the visual identification of regions in which the combined behavior of the queried variables is statistically more important to the inquiry. Based on the univariate distribution functions, the query solution is then segmented into different subregions in which the distribution of different variables is more important in defining the queries' solution. Figure 4b shows an example of the segmentation of a query defining the eye of a hurricane. Segmentation of the query solution based on the univariate density functions reveals three regions in which the query's joint distribution is dominated by the influence of pressure (blue), velocity (green), and temperature (red). The comparison of a query's univariate distributions to the corresponding distributions, restricted to the segmented regions, can help identify parameters for further query-refinement.

As illustrated by the above example, understanding the structure of a query defined by the trends and interactions of variables within the query's solution, can provide important insight to help with the refinement of queries. To this end, Gosink et al. [16] proposed the use of univariate cumulative distribution functions restricted to the solution space of the query to identify principle level sets for all variables that are deemed the most relevant to the query's solution. Additional derived scalar fields, describing the local pairwise correlation between two variables in physical space are then used to identify a pairwise variable interactions. Visualization of these correlation fields, in conjunction with principle isosurfaces, then enables the identification of statistically important interactions and trends between any three variables.

The manual, query-driven exploration of extremely large data sets enables the user to build and test new hypotheses. However, manual exploration is often a time-consuming and complex process and is, therefore, not well-suited for the processing of large collections of scientific data. Combining the QDV concept with methods to automate the definition of advanced queries to extract specific features of interest helps to support the analysis of large data collection. For example, in the context of plasma-based particle accelerators, many analyses rely on the ability to accurately define the subset of particles that form the main particle beams of interest. To this effect, Section 4.2 describes an efficient query-based algorithm for the automatic detection of particle beams [27]. Combining the automatic query-based beam analysis with advanced visualization supports an efficient analysis of complex plasma-based accelerator simulations, as shown in Figure 4c.

4 Applications of Query-Driven Visualization

QDV is among the small subset of techniques that can address both large and highly complex data. QDV is an efficient, feature-focused analysis approach that has a wide range of applications. The case studies that follow all make use of supercomputing platforms to perform QDV and analysis on data sets of unprecedented size. One theme across all these studies is that QDV reduces visualization and analysis processing time from hours or days, to seconds or minutes.

The study results presented in Section 4.1 aim to accelerate analysis within the context of forensic cybersecurity [3,30]. Two high energy physics case studies are presented in Section 4.2, both of which are computational experiments aimed at designing next-generation particle accelerators, such as: a free-electron laser (Section 4.2.1) and a plasma-wakefield accelerator (Section 4.2.2).

4.1 Applications in Forensic Cybersecurity

Modern forensic analytics applications, like network traffic analysis, consist of hypothesis testing, knowledge discovery, and data mining on very large data sets. One key strategy to reduce the time-to-solution is to be able to quickly focus analysis on the subset of data relevant for a given analysis. This case study, presented in earlier work [3,30], uses a combination of supercomputing platforms for parallel processing, high performance indexing for fast searches, and user interface technologies for specifying queries and examining query results.

The problem here is to find and analyze a distributed network scan attack buried in one year's worth of network flow data, which consists of 2.5 billion records. In a distributed scan, multiple distributed hosts systematically probe for vulnerabilities on ports of a set of target hosts. The traditional approach, consisting of command line scripts and using ASCII files, could have required many weeks of processing time. The approach presented below reduces this time to minutes.

Data: In this experiment, the data set consists of network connection data for a period of 42 weeks—a total of 2.5 billion records—acquired from a Bro system [24] running at a large

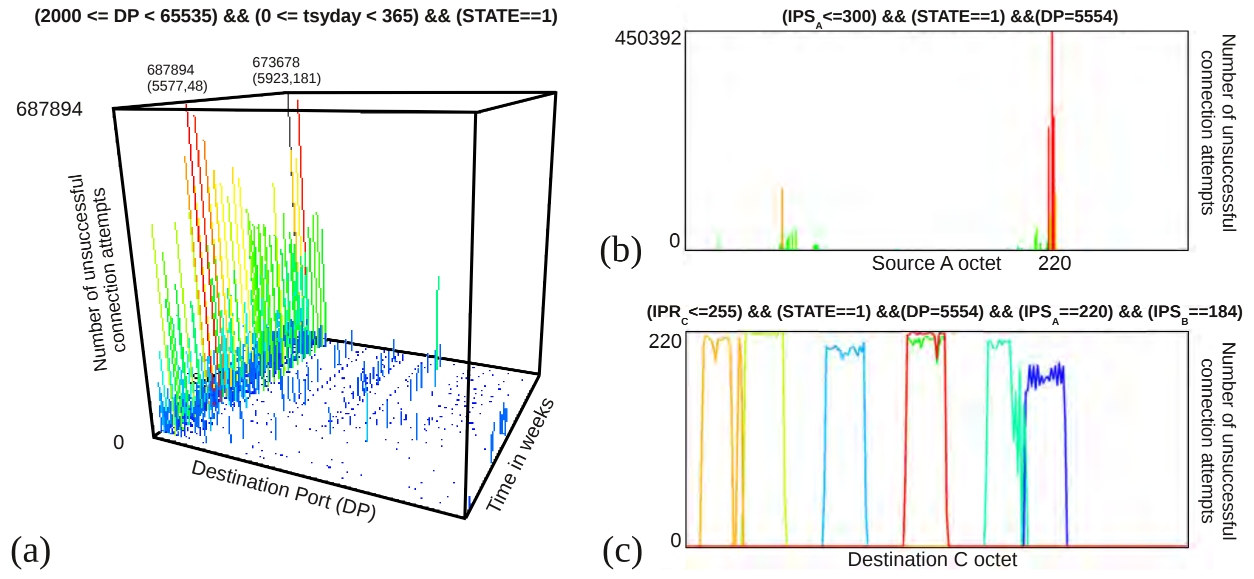


Figure 5: Histograms showing the number of unsuccessful connection attempts (with radiation excluded): (a) on ports 2000 to 65535 over a 42-week period, indicating high levels of activity on port 5554 during the 7th week; (b) per source A octet during the 7th week on port 5554, indicating suspicious activity from IPs with a 220 A octet; (c) per destination C octet scanned by seven suspicious source hosts (color), indicating a clear scanning pattern. Image source: Bethel et al., 2006 [3].

supercomputing facility. The data set contains for each record, the “standard” set of connection variables, such as source and destination IP addresses, ports, connection duration, etc. The data is stored in flat files, using an uncompressed binary format for a total size of about 281GB. The IP addresses are split into four octets, $A : B : C : D$, to improve query performance. For instance, IPS_A refers to the class A octet of the source IP address. The FastBit bitmap indices used to accelerate data queries require a total of $\approx$ 78.6GB of space.

Interactive Query Interface: The study’s authors showed a histogram-based visualization and query interface to facilitate the exploration of network traffic data. After loading the metadata from a file, the system invokes FastBit to generate coarse temporal resolution histograms containing, for example, counts of variables over the entire temporal range at a weekly resolution. The user then refines the display by drilling into a narrower temporal range and at a finer temporal resolution. The result of such a query is another histogram, which is listed as a new variable in the user interface. A more complex query may be formed by the “cross-product” of histograms of arbitrary numbers of data variables, and may be further qualified with arbitrary, user-specified conditions.

One advantage of such a histogram-centric approach is that it allows for an effective iterative refinement of queries. This iterative, multiresolution approach, which consists of an analyst posing different filtering criteria to examine data at different temporal scales and resolutions, enables effective context and focus changes. Rather than trying to visually analyze 42 weeks worth of data at one-second resolution all at once, the user can identify higher-level features first and then analyze those features and their subfeatures in greater detail.

Another advantage is performance. The traditional methodology in network traffic analysis consists of running command-line scripts on logfiles. Due to the nature of how FastBit stores and operates on bitmap indices, it can compute conditional histograms much more quickly than a traditional sequential scan. This idea was the subject of several different performance experiments [3, 30].

Network Traffic Analysis: The objective of this analysis example is to examine 42 weeks worth of network data to investigate an initial intrusion detection system (IDS) alert indicating a large number of scanning attempts on TCP port 5554, which is indicative of a so-called "Sasser worm." The first step, then, is to search the complete data set to identify the ports for which large numbers of unsuccessful connection attempts have been recorded by the Bro system. Figure 5a shows the counts of unsuccessful connection attempts on all ports over the entire time range at weekly temporal resolution. The chart reveals a high degree of suspicious activity in the seventh week, on destination port 5554.

Now that the suspicious activity has been confirmed and localized, the next goal is to identify the addresses of the host(s) responsible for the unsuccessful connection attempts. The splitting of IP addresses into four octets ($A : B : C : D$) enables an iterative search to determine the A, then B, C, and finally, D octet addresses of the attacking hosts. The first query then asks for the number of unsuccessful connection attempts on port 5554 during the seventh week over each address within the Class A octet. As can be seen in Figure 5b, the most suspicious activity originates from host(s), having IP addresses with a 220 A octet. Further refinements of the query, then reveal the B octet (IP=220 : 184 : x : x) and a list of C octets ($IPS_C = \{26, 31, 47, 74, 117, 220, 232\}$) from which most of the suspicious activity originates, indicating that a series of hosts on these seven IPS_C network segments may be involved in a distributed scan. The query process repeats over the D address octet to produce a complete set of IP addresses of all hosts participating in the distributed scan attack.

Once the addresses for the attacking hosts are identified, the next question is "What are the access patterns of these host addresses through destination IP addresses?" In other words, the problem is to determine the set of host addresses that are being attacked. The destination Class C octets, scanned by each of the seven source hosts—shown in Figure 5c—reveals that each of the seven participating hosts is sending traffic to about 21 or 22 contiguous Class C addresses. Further analysis of the class D octets in the context of the identified C octets, reveals that each attacking host scans through all Class D addresses for each Class C address and that each of the hosts scans a contiguous range of IPR_C's.

This case study reveals the iterative nature of exploratory analytics. The first inquiry examined all data at a coarse temporal resolution. Subsequent inquiries became more focused, looking at smaller and smaller subsets of the data. One of the previous studies estimated the time required for executing a single query to be approximately 43 hours, using traditional shell-based scripts, compared to about five seconds, using the QDV approach. Given that this case study encompasses many queries, one conclusion is that advanced exploratory analytics on very large data sets may simply not be feasible using traditional approaches.

4.2 Applications in High Energy Physics

Particle accelerators are among the most important and versatile tools in scientific discovery. They are essential to a wealth of advances in material science, chemistry, bioscience, particle physics, and nuclear physics. Accelerators also have important applications to the environment, energy, and national security. Example applications of particle accelerators include studying bacteria for bioremediation, exploring materials for solar cells, and developing accelerator-based systems to inspect cargo for nuclear contraband. Accelerator-based systems have also been proposed for accelerator-driven fission energy production and as a means to transmute nuclear waste.

Accelerators have a direct impact on the quality of people's lives through applications in medicine, such as the production of medical radioisotopes, pharmaceutical drug design and discovery, and through the thousands of accelerator-based irradiation therapy procedures that occur daily at U.S. hospitals. Given the importance of particle accelerators, it is essential that the most advanced high performance computing tools be brought to bear on accelerator R&D, and on the design, commissioning, and operation of future accelerator facilities. The following sections describe the application of QDV to the analysis of output from numerical simulations that model the behavior of electron linear particle accelerators (linacs) in Section 4.2.1 and laser plasma particle accelerators (LPA) in Section 4.2.2.

Different variations of particle-in-cell (PIC) based simulations are used to model both linacs, as well as LPA experiments [21,25]. In PIC simulations, collections of charged particles, such as electrons or protons, are modeled as computational macro-particles that can be located anywhere in the computational domain. The electromagnetic field is spatially discretized, often using a regular mesh. At each simulation step, the particles are moved under the electromagnetic forces obtained through interpolation from the fields. The current carried by the moving particles is then deposited onto the simulation grid to update the electric and magnetic fields. The data sets produced by accelerator simulations are extremely large, heterogeneous (both particles and fields), multivariate and often of highly varying spatio-temporal resolution.

The analytics process is similar to that of the previous network traffic analysis example (see Section 4.1): a scientist begins the investigation with a very large data set and formulates a set of questions, then iteratively extracts subsets of data for inspection and analysis. The pace at which knowledge is discovered is a function of the rate at which this entire process can occur. The combination of QDV and supercomputers can reduce this duty cycle from hours to seconds, thereby helping to accelerate the discovery of scientific knowledge.

4.2.1 Linear Particle Accelerator

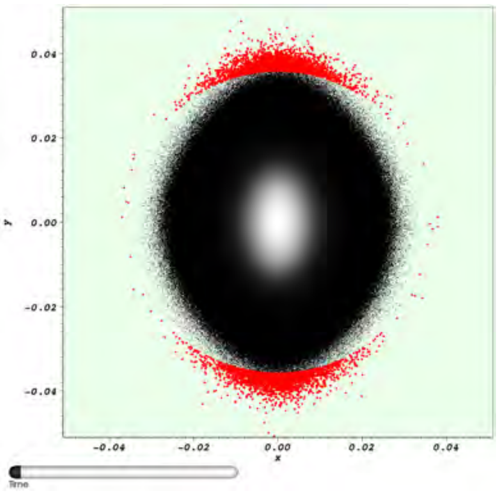
The linac case study applies QDV to the analysis of data from large-scale, high-resolution simulations of beam dynamics in linacs for a proposed next-generation x-ray, free electron laser (FEL) at LBNL [12]. PIC-based simulations on this type of accelerator require large numbers of macroparticles ($> 10^8$) to control the numerical macroparticle shot noise and to avoid overestimation of the microbunching instability, and they produce particle data sets that are massive in size [25]. The authors, Chou et al. [10], focus on applying QDV to study beam diagnostics of the transverse halo and beam core.

Data: The source data in this study was generated by the IMPACT-T simulation code [25], and consists of 720 timesteps with ≈ 1 billion particles per timestep. The 50TB data set contains nine variables describing the physical particle location and momentum. The variables x and y are the transverse location of particles. The longitudinal particle location, t, is in a relative coordinate system—that is, t identifies the particle's arrival time at the physical location z.

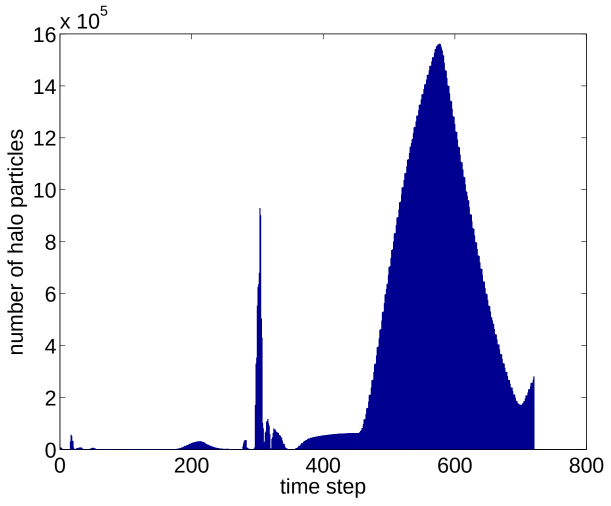
Transverse Halo: The transverse halo of a particle beam is a low-density portion of the beam usually defined as those particles beyond some specified radius or transverse amplitude in physical space, shown in Figure 6a. Controlling the beam halo is critical because the particles of the halo have the potential to reach very large transverse amplitudes, eventually striking the beam pipe and causing radioactivation of accelerator components and possibly component damage. Also, the beam halo is often the limiting factor in increasing the beam intensity to support scientific experiments. To enable a detailed analysis of this phenomenon—to determine the origin of the halo particles and to help develop halo mitigation strategies—physicists need to be able to efficiently identify and extract the halo particles.

The authors identify halo particles using the query $r > 4\sigma_r$. The derived quantity $r = \sqrt[2]{x^2 + y^2}$, which is separately computed and indexed, describes the transverse radial particle location. The transverse halo threshold is given by $\sigma_r = \sqrt[2]{\sigma_x^2 + \sigma_y^2}$, where σ_x and σ_y denote the root mean square (RMS) beam sizes in x and y, respectively. Using r for identification of halo particles is based on the assumption of an idealized circular beam cross section. To ensure that the query adapts more closely to the transverse shape of the beam, one may relax the assumption of a circular beam cross section to an elliptical beam cross section, by scaling the x and y coordinates independently by the corresponding RMS beam size, σ_x and σ_y using, for example, a query of the form $r_s^2(x, y) > 16$, with $r_s^2(x, y) = (\frac{x}{\sigma_x})^2 + (\frac{y}{\sigma_y})^2$.

Figure 6b shows the number of halo particles per timestep identified by the halo query. It shows large variations in the number of halo particles, while in particular the larger number of halo particles at later timesteps are indicative of a possible problem. In this case, the halo particles and observations of an increase in the maximum particle amplitude were found to be



(a)



(b)

Figure 6: (a) Particle density plot (gray) and selected halo-query particles (red) for timestep 20 of the simulation. (b) Plot showing the number of halo particles per timestep in a 50TB electron linear particle accelerator data set. Image source: Chou et al., 2011 [10].

due to a mismatch in the beam as it traveled from one section of the accelerator to the next. These type of query-based diagnostics provide accelerator designers with evidence that further improvement of the design may be possible, and also provides quantitative information that is useful for optimizing the design to reduce halo formation and beam interception with the beam pipe, which will ultimately improve accelerator performance.

Core: Creating beams with good longitudinal beam quality, which is defined in terms of the longitudinal RMS emittance, and maintaining that beam quality during the acceleration process, is critical in certain types of accelerators involving intense electron beams. Frequently, the longitudinal "head" and "tail" of an electron bunch will be very different from its "center" region, also known as the longitudinal "core." Focusing on the longitudinal core of a particle bunch eliminates strong variations at the head and tail, which would otherwise distort the analysis results.

The authors define the core of the beam using the query $\bar{t} - 200\delta \leq t \leq \bar{t} + 200\delta$, with $\bar{t}$ being the longitudinal bunch centroid. The parameter $\delta = \frac{10^{-9}}{0.036728}$ has been provided by domain scientists and corresponds to $\approx$ 1nm; or more precisely, the time it takes an electron to travel 1nm. To identify within-core variations in particle density, uncorrelated energy spread, and transverse emittance, the core is typically further subdivided into slices for analysis purposes. In order to divide the core into, say, 400, 1nm wide slices, one could evaluate, at each timestep, multiple queries of the form $t_{min}(i) \leq t \leq t_{max}(i)$ with $t_{min} = \bar{t} - (i - 201)\delta$, $t_{max}(i) = \bar{t} - (i - 200)\delta$ and $i \in [1, 400]$. Such an analysis of all 720 timesteps of the example data set would require $288,000$ queries, further highlighting the need for efficient query methods.

Performance: The authors used the FastQuery [10] system for the parallel processing of index and query operations on the NERSC Cray XE6 supercomputing system Hopper.[1] The pre-processing to build the indexes for all timesteps of the 50TB data set required about 2 hours. Using bitmap indexing, the evaluation of the halo and core query required only 12 seconds, using 3000 cores, while practically reasonable times of around 20 seconds can be achieved using only approximately 500 compute cores. The combination of visual data exploration and efficient data management in the QDV concept enables, in this way, repeated, complex, large-scale

[1]System specification: #compute nodes $\cong$ 6,500, #cores/node=24, memory/node=32GB, filesystem=Lustre, peak I/O=25GB/s. For details see `http://www.nersc.gov/nusers/systems/hopper2/`.

query-based analysis of massive data sets, which would otherwise not be practical, with respect to both time as well as computational cost.

4.2.2 Laser Plasma Particle Accelerator

Plasma-based particle accelerators (LPAs) [13] use a short ($\leq$ 100fs), ultrahigh intensity ($\geq 10^{18} W/cm^2$) laser pulse to drive waves in a plasma. Electrons in a hydrogen plasma are displaced by the radiation pressure of the laser pulse, while the heavier ions remain stationary. This displacement of the electrons in combination with the space-charge restoring force of the ions, drives a wave (wake) in the plasma. Similar to a surfer riding a wave, electrons that become trapped in the plasma wave are accelerated by the wave to high energy levels. LPAs can achieve electric and magnetic fields thousands of times stronger than conventional accelerators, enabling the acceleration of particles to high energy levels within very short distances (centimeters to meters). Researchers at the LOASIS[2] program have demonstrated high-quality electron beams at 0.1 to 1 GeV using millimeter to centimeter long plasmas [15, 20].

One central challenge in the analysis of large LPA simulation data arises from the fact that, while large numbers of particles are required for an accurate simulation, only a small fraction of the particles are accelerated to high energies and subsequently form particle features of interest. During the course of a simulation, multiple high-energy particle beams may form and additional particle bunches may appear in secondary periods of the plasma wave. In this context, scientists are particularly interested in the analysis of the main particle beam. This section discusses query-driven analysis of LPA simulations used to extract, analyze, and compare high-energy particle beams.

Data: This case study uses data sets produced by VORPAL [21], an electromagnetic PIC-based simulation code. Accurate modeling of LPAs is computationally expensive, in particular, due to large differences in scale, e.g., length of the plasma versus the laser wavelength. To save computational resources and storage space, VORPAL employs a moving-window simulation approach. In this method, only a window around the laser pulse is simulated at each timestep, and it is moved along at the speed of light, as the laser propagates through the plasma. The output data contains particle position (x, y, z), particle momentum (px, py, pz), particle weight (wt), and particle identifier (id). The data is indexed using FastBit to accelerate the evaluation of range queries, id-based equality queries, and the computation of conditional histograms.

Interactive Query-Driven Data Exploration: To gain a deeper understanding of the acceleration process, physicists need to address a number of complex questions, such as: Which particles become accelerated? How are particles trapped and accelerated by the plasma wave? How are the beams of highly accelerated particles formed? And, how do the particle beams evolve over time? The case study tackles these questions via an interactive, query-driven visual exploration approach for histogram-based parallel coordinates (see Section 3.1) and high performance scientific visualization [28].

To identify those particles that were accelerated, the authors, who included an accelerator physicist, performed an initial threshold selection in px at a late timestep of the simulation. This initial selection restricts the analysis to a small set of particles with energies above the base acceleration of the plasma wave. Figure 7a shows an example of a parallel coordinates plot of such a selection (gray).

Based on the results of the first query, the selection was further refined to select the main particle beams of interest. The authors first increase the px threshold to extract the particles of highest energy. This initial refinement often results in the selection of multiple beam-like features trapped in different periods of the plasma wave.

As illustrated in Figure 7a (red lines), to separate the different particle beams and to extract the main particle beam, the selection is then often further refined through range queries in the longitudinal coordinate x and transverse coordinates y and z. In the selection process, parallel coordinates provide interactive feedback about the structure of the selection, allowing for fast

[2]Lasers, Optical Accelerator Systems Integrated Studies (LOASIS) Program at LBNL; `http://loasis.lbl.gov/`

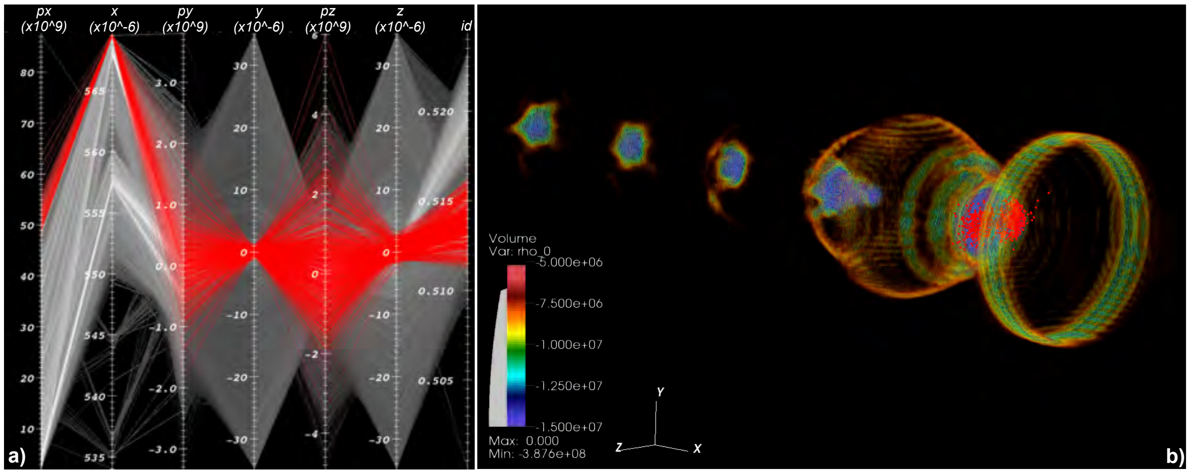


Figure 7: QDV of large 3D plasma-based particle acceleration data containing $\approx 90 \times 10^6$ particles per timestep. On the left (a), parallel coordinates of timestep $t = 12$ showing: (1) all particles above the base acceleration level of the plasma wave (Query: $px > 2 \times 10^9$) (gray) and (2) a set of particles that form a compact beam in the first wake period following the laser pulse (Query: $(px > 4.856 \times 10^{10})\text{AND}(x > 5.649 \times 10^{-4})$)(red). On the right (b), volume rendering of the plasma density illustrating the 3D structure of the plasma wave. The selected beam particles are shown in addition in red. Image source: Rübel et al., 2008 [28].

identification of outliers and different substructures of the selection. High performance scientific visualization methods are then used to validate and further analyze the selected particles Figure 7b.

The next step is to trace, or follow, and analyze the high-energy particles through time. Particle tracing is used to detect the point in time when the beam reaches its peak energy and to assess the quality of the beam. Tracing the beam particles further back in time, to the point at which the particles enter the simulation window, supports analysis of the injection, beam formation, and beam evolution processes. Figure 3c shows an example in which parallel coordinates are used to compare the acceleration behavior of two particle beams. Based on the information from different individual timesteps, a user may refine a query, to select, for example, beam substructures that are visible at different discrete timesteps. Rübel et al. [28] demonstrated that using FastBit-accelerated equality queries, the time needed for tracing beam particles can be reduced from many hours to less than a second.

Automatic Query-Based Beam Detection: Interactive query-driven visual data exploration is effective in that it provides great flexibility and supports detailed data analysis, allowing scientist to define and validate new hypotheses about the data and the physical phenomena being modeled. However, manual detection and extraction of the acceleration features of interest, such as particle beams, is time consuming. To support the analysis of large collections of accelerator simulations, efficient methods are needed for automatic detection and extraction of particle beams.

Rübel et al. [27] described an efficient query-based algorithm for automatic detection of particle beams. At a single timestep, the analysis proceeds as follows. First, the algorithm computes a 3D conditional histogram of (x, y, px) space, restricted to high-energy particles selected by the query $px > 1e10$. Using a two-stage segmentation process, the algorithm first identifies the region in the physical space (x, y), containing the highest energy particles. The so-defined spatial selection is then refined using a 3D region-growing-based segmentation centered around the highest-density histogram bin. Histogram-based bin-queries (see Section 2.1) are then used to extract the particles belonging to the so-identified feature. The identified per-timestep features are then traced over time and refined through additional per-timestep segmentations.

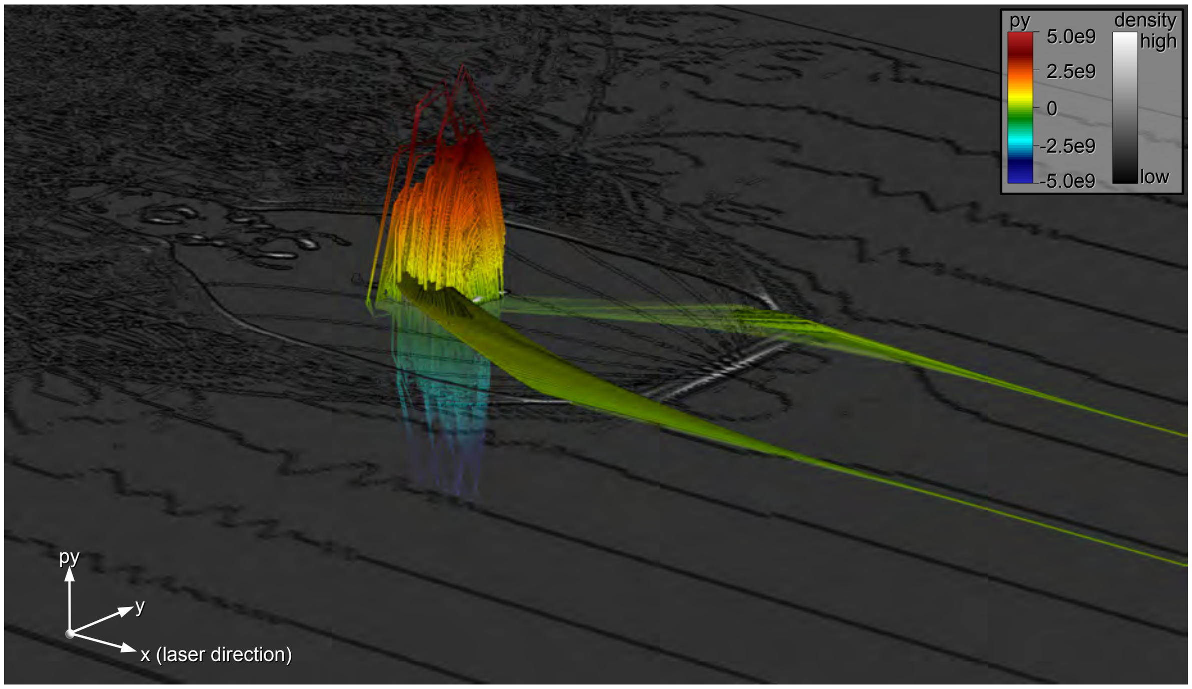


Figure 8: Visualization of the relative traces of a particle beam in a laser plasma accelerator. The traces show the motion of the beam particles relative to the laser pulse. The xy-plane shows isocontours of the particle density at the timepoint when the beam reaches its peak energy, illustrating the location of the beam within the plasma wave. The positive vertical axis and color of the particle traces show the particle momentum in the transverse direction, py. The image shows particles being injected from the sides and oscillating while being accelerated. Image source: Rübel et al., 2009 [27].

Finally, the particles of interest are traced over time using a series of ID-based equality queries, and additional particle-to-feature distance fields are computed. In this query-based analysis process, the computation of conditional histograms, the evaluation of histogram-based bin queries, and the identifier-based particle tracing are accelerated through FastBit.

Even for a large 3D data set (approximately 620GB) consisting of 26 timesteps with approximately 230×10^6 particles per timestep, the analysis requires only about 3 minutes in serial, which is already much less than what a single histogram-based bin-query would require without using FastBit. As shown in Figure 2b, the analysis has to evaluate tens of bin queries, 52 threshold and equality queries. Combining the query-based beam analysis with advanced visualization supports automated analysis and comparison of particle beams in complex LPA simulations (see Fig. 4c). Figure 8 shows as an example a visualization of the injection and acceleration process of an automatically detected particle beam.

5 Conclusion

Query-driven visualization and analysis is a blend of technologies that limit visualization and analysis processing to the subset of data that is "interesting." The premise is that, in any given investigation, only a small fraction of a large data set is of interest.

QDV is useful for fast visual exploration of extremely large data sets. It has been shown to be useful to accelerate and automate complex feature detection tasks, in particular, when combined with other analysis methods, such as unsupervised learning. As a general concept, QDV is very flexible and can be easily integrated with a large range of other methods. Advanced index/query systems (e.g., FastQuery) and integration of these data interfaces with advanced visualization systems (such as VisIt), make QDV-based analysis much more accessible to the scientific community.

QDV is among the small subset of techniques that can address visualization and analysis of large and highly complex data. The case studies in this chapter, taken from forensic cybersecurity and high-energy physics applications, make use of supercomputing platforms to perform QDV and analysis on data sets of unprecedented sizes. Other recent work in QDV, not discussed here, applies the concepts to diverse application areas, like computational finance, climate modeling and analysis, magnetically confined fusion, and astrophysics. One theme across all of these studies is that QDV reduces visualization and analysis processing time from hours or days to seconds or minutes.

References

[1] G. Antoshenkov. Byte-aligned Bitmap Compression. Technical report, Oracle Corp., 1994. U.S. Patent number 5,363,098.

[2] Rudolf Bayer and Edward McCreight. Organization and Maintenance of Large Ordered Indexes. *Acta Informatica*, 3:173–189, 1972.

[3] E. Wes Bethel, Scott Campbell, Eli Dart, Kurt Stockinger, and Kesheng Wu. Accelerating Network Traffic Analysis Using Query-Driven Visualization. In *Proceedings of 2006 IEEE Symposium on Visual Analytics Science and Technology*, pages 115–122. IEEE Computer Society Press, October 2006. LBNL-59891.

[4] E. Wes Bethel, Hank Childs, and Charles Hansen, editors. *High Performance Visualization—Enabling Extreme-Scale Scientific Insight.* Chapman & Hall, CRC Computational Science. CRC Press/Francis–Taylor Group, Boca Raton, FL, USA, November 2012. `http://www.crcpress.com/product/isbn/9781439875728`.

[5] Peter A. Boncz, Marcin Zukowski, and Niels Nes. MonetDB/X100: Hyper-Pipelining Query Execution. In *Proceedings of the Biennial Conference on Innovative Data Systems Research (CIDR)*, pages 225–237, January 2005.

[6] Peer-Timo Bremer, Gunther H. Weber, Valerio Pascucci, Marcus S. Day, and John B. Bell. Analyzing and Tracking Burning Structures in Lean Premixed Hydrogen Flames. *IEEE Transactions on Visualization and Computer Graphics*, 16(2):248–260, Mar/Apr 2010. LBNL-2276E.

[7] Paul G. Brown. Overview of sciDB: Large Scale Array Storage, Processing and Analysis. In *Proceedings of the 2010 ACM SIGMOD International Conference on Management of Data*, SIGMOD '10, pages 963–968, Indianapolis, IN, USA, 2010.

[8] C. Y. Chan and Y. E. Ioannidis. Bitmap Index Design and Evaluation. In *Proceedings of the 1998 ACM SIGMOD International Conference on Management of Data*, SIGMOD '98, pages 355–366, New York, NY, USA, June 1998. ACM.

[9] C. Y. Chan and Y. E. Ioannidis. An Efficient Bitmap Encoding Scheme for Selection Queries. In *SIGMOD*, Philadelphia, PA, USA, June 1999. ACM Press.

[10] Jerry Chou, Mark Howison, Brian Austin, Kesheng Wu, Ji Qiang, E. Wes Bethel, Arie Shoshani, Oliver Rübel, Prabhat, and Rob D. Ryne. Parallel Index and Query for Large Scale Data Analysis. In *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, SC '11, pages 30:1–30:11, Seattle, WA, USA, November 2011.

[11] Douglas Comer. The Ubiquitous B-Tree. *Computing Surveys*, 11(2):121–137, 1979.

[12] J. N. Corlett, K. M. Baptiste, J. M. Byrd, P. Denes, R. J. Donahue, L. R. Doolittle, R. W. Falcone, D. Filippetto, J. Kirz, D. Li, H. A. Padmore, C. F. Papadopoulos, G. C. Pappas, G. Penn, M. Placidi, S. Prestemon, J. Qiang, A. Ratti, M. W. Reinsch, F. Sannibale, D. Schlueter, R. W. Schoenlein, J. W. Staples, T. Vecchione, M. Venturini, R. P. Wells, R. B. Wilcox, J. S. Wurtele, A. E. Charman, E. Kur, and A. Zholents. A Next Generation Light Source Facility at LBNL. In *Proceedings of PAC 2011*, New York, NY, USA, April 2011.

[13] E. Esarey, C. B. Schroeder, and W. P. Leemans. Physics of Laser-Driven Plasma-Based Electron Accelerators. *Reviews of Modern Physics*, 81:1229–1285, 2009.

[14] V. Gaede and O. Günther. Multidimension Access Methods. *ACM Computing Surveys*, 30(2):170–231, 1998.

[15] C. G. R. Geddes, Cs. Toth, J. van Tilborg, E. Esarey, C. B. Schroeder, D. Bruhwiler, C. Nieter, J. Cary, and W. P. Leemans. High-Quality Electron Beams from a Laser Wakefield Accelerator Using Plasma-Channel Guiding. *Nature*, 438:538–541, 2004. LBNL-55732.

[16] Luke Gosink, John C. Anderson, E. Wes Bethel, and Kenneth I. Joy. Variable Interactions in Query-Driven Visualization. *IEEE Transactions on Visualization and Computer Graphics (Proceedings of Visualization 2007)*, 13(6):1400–1407, November/December 2007. LBNL-63524.

[17] Luke Gosink, John Shalf, Kurt Stockinger, Kesheng Wu, and E. Wes Bethel. HDF5-FastQuery: Accelerating Complex Queries on HDF Datasets using Fast Bitmap Indices. In *Proceedings of the 18th International Conference on Scientific and Statistical Database Management*, pages 149–158. IEEE Computer Society Press, July 2006. LBNL-59602.

[18] Luke J. Gosink, Christoph Garth, John C. Anderson, E. Wes Bethel, and Kenneth I. Joy. An Application of Multivariate Statistical Analysis for Query-Driven Visualization. *IEEE Transactions on Visualization and Computer Graphics*, 17(3):264–275, 2011. LBNL-3536E.

[19] Alfred Inselberg. *Parallel Coordinates Visual Multidimensional Geometry and Its Applications.* Springer-Verlag, Secaucus, NJ, USA, 2008.

[20] W. P. Leemans, B. Nagler, A. J. Gonsalves, Cs. Toth, K. Nakamura, C. G. R. Geddes, E. Esarey, C. B. Schroeder, and S. M. Hooker. GeV Electron Beams from a Centimetre-Scale Accelerator. *Nature Physics*, 2:696–699, 2006.

[21] C. Nieter and J. R. Cary. VORPAL: A versatile plasma simulation code. *Journal of Computational Physics*, 196(2):448–473, 2004.

[22] Matej Novotný and Helwig Hauser. Outlier-Preserving Focus+Context Visualization in Parallel Coordinates. *IEEE Transactions on Visualization and Computer Graphics*, 12(5):893–900, 2006.

[23] P. O'Neil. Model 204 Architecture and Performance. In *Proceedings of the 2nd International Workshop on High Performance Transaction Systems*, pages 40–59, Asilomar, CA, USA, September 1987. Springer-Verlag.

[24] Vern Paxson. Bro: A System for Detecting Network Intruders in Real-Time. *Computer Networks*, 31(23-24):2435–2463, 1999.

[25] J. Qiang, R. D. Ryne, M. Venturini, and A. A. Zholents. High Resolution Simulation of Beam Dynamics in Electron Linacs for X-Ray Free Electron Lasers. *Physical Review*, 12(10):100702–1–100702–11, 2009.

[26] Doron Rotem, Kurt Stockinger, and Kesheng Wu. Optimizing I/O Costs of Multi-dimensional Queries Using Bitmap Indices. In *Proceedings of the 16th International Conference on Database and Expert Systems Applications*, DEXA'05, pages 220–229, 2005.

[27] Oliver Rübel, Cameron G. R. Geddes, Estelle Cormier-Michel, Kesheng Wu, Prabhat, Gunther H. Weber, Daniela M. Ushizima, Peter Messmer, Hans Hagen, Bernd Hamann, and E. Wes Bethel. Automatic Beam Path Analysis of Laser Wakefield Particle Acceleration Data. *IOP Computational Science & Discovery*, 2(1):015005, November 2009. LBNL-2734E.

[28] Oliver Rübel, Prabhat, Kesheng Wu, Hank Childs, Jeremy Meredith, Cameron G. R. Geddes, Estelle Cormier-Michel, Sean Ahern, Gunther H. Weber, Peter Messmer, Hans Hagen, Bernd Hamann, and E. Wes Bethel. High Performance Multivariate Visual Data Exploration for Extemely Large Data. In *Supercomputing 2008 (SC08)*, Austin, Texas, USA, November 2008. LBNL-716E.

[29] Arie Shoshani and Doron Rotem, editors. *Scientific Data Management: Challenges, Technology, and Deployment.* Chapman & Hall/CRC Press, Boca Raton, FL, USA, 2010.

[30] Kurt Stockinger, E. Wes Bethel, Scott Campbell, Eli Dart, and Kesheng Wu. Detecting Distributed Scans Using High-Performance Query-Driven Visualization. In *SC '06: Proceedings of the 2006 ACM/IEEE Conference on High Performance Computing, Networking, Storage and Analysis.* IEEE Computer Society Press, November 2006. LBNL-60053.

[31] Kurt Stockinger, John Shalf, E. Wes Bethel, and Kesheng Wu. DEX: Increasing the Capability of Scientific Data Analysis Pipelines by Using Efficient Bitmap Indices to Accelerate Scientific Visualization. In *Proceedings of Scientific and Statistical Database Management Conference (SSDBM)*, pages 35–44, Santa Barbara, CA, USA, June 2005. LBNL-57203.

[32] Kurt Stockinger, John Shalf, Kesheng Wu, and E. Wes Bethel. Query-Driven Visualization of Large Data Sets. In *Proceedings of IEEE Visualization 2005*, pages 167–174. IEEE Computer Society Press, October 2005. LBNL-57511.

[33] The HDF Group. HDF5 User Guide. `http://hdf.ncsa.uiuc.edu/HDF5/doc/H5.user.html`, 2010.

[34] Unidata. The NetCDF Users' Guide. `http://www.unidata.ucar.edu/software/netcdf/docs/netcdf/`, 2010.

[35] Michelle Q. Wang Baldonado, Allison Woodruff, and Allan Kuchinsky. Guidelines for Using Multiple Views in Information Visualization. In *Proceedings of the Working Conference on Advanced Visual Interfaces*, AVI '00, pages 110–119, New York, NY, USA, 2000. ACM.

[36] K. Wu, W. Koegler, J. Chen, and A. Shoshani. Using Bitmap Index for Interactive Exploration of Large Datasets. In *Proceedings of the 15th International Conference on Scientific and Statistical Database Management*, SSDBM '03, pages 65–74. IEEE Computer Society Press., July 2003.

[37] Kesheng Wu, Sean Ahern, E. Wes Bethel, Jacqueline Chen, Hank Childs, Estelle Cormier-Michel, Cameron Geddes, Junmin Gu, Hans Hagen, Bernd Hamann, Wendy Koegler, Jerome Lauret, Jeremy Meredith, Peter Messmer, Ekow Otoo, Victor Perevoztchikov, Arthur Poskanzer, Prabhat, Oliver Rübel, Arie Shoshani, Alexander Sim, Kurt Stockinger, Gunther Weber, and Wei-Ming Zhang. FastBit: Interactively Searching Massive Data. In *SciDAC 2009*, 2009. LBNL-2164E.

[38] Kesheng Wu, Ekow Otoo, and Arie Shoshani. On the Performance of Bitmap Indices for High Cardinality Attributes. In *Proceedings of the Thirtieth International Conference on Very large Data Bases—Volume 30*, VLDB '04, pages 24–35. VLDB Endowment, 2004.

[39] Kesheng Wu, Ekow Otoo, and Arie Shoshani. Optimizing Bitmap Indices with Efficient Compression. *ACM Transactions on Database Systems*, 31:1–38, 2006.

[40] Kesheng Wu, Ekow Otoo, and Kenji Suzuki. Optimizing Two-Pass Connected-Component Labeling Algorithms. *Pattern Analysis & Applications*, 12(2):117–135, 2009. `http://www.springerlink.com/index/B67258V347158263.pdf`.

[41] Kesheng Wu, Arie Shoshani, and Kurt Stockinger. Analyses of Multi-Level and Multi-Component Compressed Bitmap Indexes. *ACM Transactions on Database Systems*, 35(1):1–52, 2010. `http://doi.acm.org/10.1145/1670243.1670245`.

[42] Kesheng Wu, Rishi R Sinha, Chad Jones, Stephane Ethier, Scott Klasky, Kwan-Liu Ma, Arie Shoshani, and Marianne Winslett. Finding Regions of Interest on Toroidal Meshes. *Computational Science & Discovery*, 4(1):015003, 2011.

[43] Kesheng Wu, Kurt Stockinger, and Arie Shosani. Breaking the Curse of Cardinality on Bitmap Indexes. In *Proceedings of the 20th International Conference on Scientific and Statistical Database Management*, SSDBM '08, pages 348–365. Springer-Verlag, 2008.